RESOURCE-AWARE AND ROBUST IMAGE PROCESSING FOR INTELLIGENT SENSOR SYSTEMS

A Dissertation
Presented to
The Academic Faculty

By

Jong Hwan Ko

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Electrical and Computer Engineering

Georgia Institute of Technology

August 2018

RESOURCE-AWARE AND ROBUST IMAGE PROCESSING FOR INTELLIGENT SENSOR SYSTEMS

Approved by:

Dr. Saibal Mukhopadhyay, Advisor
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Sudhakar Yalamanchili
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Arijit Raychowdhury
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Matthai Philipose
Department of Computer Science and Engineering
University of Washington

Dr. Tushar Krishna
School of Electrical and Computer Engineering
Georgia Institute of Technology

Date Approved: April 20, 2018

ACKNOWLEDGEMENTS

First and foremost, I would like to acknowledge God, his Son, and the Holly Spirit. I would like to thank for all the blessings, the hope, and the grace that have been poured down in life.

I would like to express my highest gratitude to my advisor, Professor Saibal Mukhopadhyay, for the full spectrum of supports he has provided during my doctoral studies. His invaluable guidance and research vision have deeply inspired me and significantly improved the quality of my research work. Beside being the doctoral advisor, he also spent many hours in mentoring me about choosing the right career path and different aspects of life for which I will remain thankful.

My sincere thanks also goes to my dissertation committee members, Professor Sudhakar Yalamanchili, Professor Arijit Raychowdhury, Professor Tushar Krishna, and Dr. Matthai Philipose for investing their valuable time and effort in shaping my dissertation.

I am grateful to former and current GREEN Lab members, Dr. Kwanyeob Chae, Dr. Amit Trivedi, Dr. Boris Alexandrov, Dr. Denny Lie, Dr. Wen Yueh, Dr. Zakir Khondker, Dr. Sergio Carlo, Dr. Jae Ha Kung, Dr. Monodeep Kar, Dr. Duckhwan Kim, Dr. Mohammad Faisal Amir, Dr. Dhongwoon Jang, Dr. Taesik Na, Arvind Singh, Yun Long, Burhan Mudassar, Edward Lee, Venkata Chaitanya Krishna Chekuri, Minah Lee, Nihar Dasari, Priyabrata Saha, Nikhil Chwala, and Xueyuan She for the support and constant encouragement that they have offered me.

I would also like to extend my special thanks to Dr. Kwanyeob Chae, Dr. Duckhwan Kim, Dr. Jae Ha Kung, and Dr. Zakir Khondker who had mentored me through my early years of PhD. I would like to acknowledge Dr. Zakir Khondker, Dr. Jae Ha Kung, Dr. Duckhwan Kim, Dr. Mohammad Faisal Amir, Dr. Taesik Na, Arvind Singh, and Burhan Mudassar who have contributed to the completion of my thesis.

I am especially thankful to Taesik Na for being a great friend and colleague. I will for-

ever remain grateful for his inspirations, suggestions and confidence in me which motivated me to undertake the challenges of getting a PhD.

I thankfully acknowledge the Agency for Defense Development (ADD) in Korea for the support during my PhD study. I also thank colleagues in ADD for shaping the dream of my PhD study and providing the opportunity to pursue it.

Finally, my greatest thanks to my wife Min Ah Kang, and my children Bomin and Daniel Ko for their continuous love, care, and support. I also thank my parents and sister for supporting my back in all aspects of my life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

The objective of this research is to design resource-aware and robust image processing algorithms, system architecture, and hardware implementation for intelligent image sensor systems in the Internet-of-Things (IoT) environment. The research explores the design of a wireless image sensor system with low-overhead pre-processing, which is integrated with a reconfigurable energy-harvesting image sensor array to implement a self-powered image sensor system. For reliable delivery of region-of-interest (ROI) under dynamic environment, the research designs low-power moving object detection with enhanced noise robustness. The system energy is further optimized by a low-power ROI-based coding scheme, whose parameters are dynamically controlled by a low-power rate controller to minimize required buffer size with minimum computational overhead. To enable machine learning based intelligent image processing at the IoT edge devices, the research proposes resource-efficient neural networks. The storage demand is reduced by compressing the neural network weights with an adaptive image encoding algorithm, and the computation demand is optimized by mapping the entire network parameters and operations into the frequency domain. To further improve the energy-efficiency and throughput of the edge device, the research explores inference partitioning of a DNN between the edge and the host platforms.

CHAPTER 1
INTRODUCTION

With the increasing demand of ubiquitous sensing and remote monitoring, wireless image sensor systems are expected to become essential components of the Internet of Things (IoT) environment. As these sensor nodes are usually operated under stringent resource constraints and dynamic variations of operating condition, a critical goal in a design of a sensor node is to enhance resource efficiency and robustness in delivering visual information.

Conventional image sensor systems have focused on the capabilities of wireless sensor nodes to capture, process, and transmit visual information to the base station, while leaving the task of video analysis to human operators [1]. In this configuration, delivering images with higher perceptual quality to human operators is critical. Therefore, the sensor node design requires exploring resource-quality scalability to optimally allocate limited resources for better visual information.

As human visual attention focuses on the Region-of-Interest (ROI) in an image, ROI-based processing (ROI detection and ROI coding) is a common design approach for better energy-quality scalability of sensor nodes [2]. In surveillance applications where the ROIs are defined as the regions with moving objects, sensor systems usually incorporate moving object detection methods to optimally allocate the resources while preserving the quality of moving objects. Once the ROI is determined, ROI coding allocates the available data rate to the ROI/non-ROI for higher ROI quality and graceful degradation of the non-ROI quality [3]. One of the biggest challenges to the ROI detection is dynamic environmental noise that can be falsely detected as moving objects. Another challenge is the stringent resource constraints such as energy, area, and transmission bandwidth. Therefore, the ROI-based processing methods should be robust to the noise as well as efficient in terms of memory

and computation demand.

The optimization of video quality and energy consumption also involves variation in a wireless channel condition. When the channel condition worsens, a wireless transmitter usually enhances channel reliability at the cost of a lower channel data rate or higher signal power, which results in an increase in transmission energy or degradation of the quality. Therefore, an optimal energy-quality tradeoff under varying channel conditions requires system-wide feedback control that adaptively tunes the target data rate. Once the target data rate is determined, the remaining challenge is to guarantee the data volume generated by an image processor match the target data rate, since the mismatch imposes energy/area overhead due to large buffer requirement. However, the encoding rate can vary due to the variable content of video frames. Consequently, to minimize the latency and buffer requirement, there is a need for an on-line rate controller that matches the encoding rate with the transmission data rate.

In remote sensing and military surveillance applications of image sensor systems, the sensor systems are usually deployed in areas where human intervention for battery replacement is a costly operation [4]. Therefore, it is highly desirable that the sensor node can harvest energy to sustain longer with a limited battery or even power itself without a battery. To enable a truly self-powered system, an energy harvesting device should be an on-chip module such as an image sensor array, and harvested energy should support a complete image sensor including an image processor, memory, a power management unit, and a transmission controller.

In addition to conventional model-based image processing algorithms, there is a growing interest in complex deep neural networks (DNNs) for intelligent computer vision tasks including image classification, pattern recognition, and gesture detection [5][6][7]. To eliminate the need for human involvement in the human operator based surveillance system, deep neural networks can be adopted at the host platform to build an intelligent surveillance system. In such a system configuration, the most critical performance metric is detec-

tion/classification accuracy of the neural network. Therefore, ROI-based processing of the image sensor node should be designed to process the video in a way that the neural network achieves higher accuracy.

DNNs are also widely adopted at the IoT edge devices to enable more intelligence. However, the biggest challenge is their storage demand and computational complexity. As DNNs contain a large number of synaptic weights, the memory demand is a key challenge for application of DNNs, especially for memory-constrained platforms such as mobile systems. Another bottleneck of DNNs is the computation demand, mostly due to their large number of multiplications in convolution layers. Therefore, reducing the storage and computation demand of neural networks is critical, specifically, to support in-field and on-chip training and inference.

The goal of this research is to design an intelligent image sensor system through resource-aware and robust image processing algorithms and deep neural networks. This goal is achieved by first designing an low-power wireless sensor system with ROI detection and coding. The system is integrated with an energy-harvesting image sensor array to implement a self-powered image sensor system. The moving object detection method is enhanced to have robustness to the environmental noise. Based on the ROI information, the system energy is further optimized by a low-power ROI-based coding scheme. The research also investigates how the energy-quality scalability of ROI-based processing is translated into the energy-accuracy scalability. The ROI-based processing parameters are dynamically controlled by a low-power rate controller to minimize required buffer size with minimum computational overhead.

To enable deep learning based intelligent image processing at the IoT edge devices with limited hardware resource, the research proposes neural network design with lower storage and computation demand. The storage demand is reduced by compressing the neural network weight, and the computation demand is optimized by mapping the network into the frequency domain. The research also explores inference partitioning of a DNN

between the edge and the host platforms to improve the energy-efficiency and throughput of the edge device.

The rest of the thesis is organized as follows: In Chapter 2, the detailed background and literature survey is presented. Chapter 3 presents the design and implementation of low-power wireless image sensor system. Chapter 4 presents energy-efficient and robust image processing through RoI-based processing. Chapter 5 introduces on-line adaptation of target data rate and encoding data rate according to the variations in wireless channel and input video characteristics. In Chapter 6, techniques for memory- and computationally-efficient deep neural networks are presented. Chapter 7 introduces partitioning of neural network inference for resource-efficient inference at the IoT edge devices. Finally, Chapter 8 describes the key research contributions and future research directions.

CHAPTER 2

BACKGROUND

2.1 Image Sensor Systems with ROI-based Processing

To enhance the efficiency of wireless video sensor systems, several studies have suggested approaches that concentrate on the ROIs in a video. Generally, these approaches can be divided into two parts: ROI detection and ROI-based processing.

2.1.1 Moving Object Detection Methods

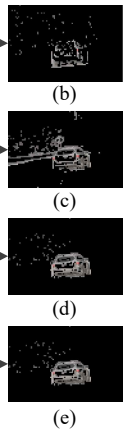
The algorithm design for ROI detection depends on how the ROI is defined. When the ROI is defined as a region with moving objects, detection of the ROI can be divided into two categories depending on the complexity and robustness: low-power moving object detection and noise-robust moving object detection.

Low-Power Moving Object Detection

Several prior studies have presented moving object detection methods for resource-constrained applications using simpler methods based on frame differencing [8], edge detection [9], and a combination of these two operations [10][11]. Frame differencing (FD) is the pixel-wise difference between two consecutive frames, followed by thresholding. Although it is of low complexity, it is subject to false detection on the frame with dynamic change in the background [10]. Edge detection (ED) has also been used for low-power systems; however, it can generate false detection since an edge image contains not only the edge of target objects but also that of background objects. Alternatively, approaches based on a combination of ED and FD have been suggested; Kim et al. proposed an algorithm based on the edge map of the inter-frame difference image (FD+ED) [10], and Ko et al. suggested

Type	Method	Memory (Bits/pixel)
Noise-robust method	GMM	120-200
	Codebook	48-192
	KDE	2424-4824
	Eigen	504
	OF	163.5-235.5
Low-power method	FD	8
	ED	0
	FD+ED	8
	ED+FD	1

(a)



(b) (c) (d) (e)

Figure 2.1: (a) Memory requirements of moving object detection methods. Example detection results under noisy condition using (b) FD, (c) ED, (d) FD+ED, and (e) ED+FD.

an algorithm based on the inter-frame difference of edge maps (ED+FD) [11]. However, as Figure 2.1(b)-(e) show, these low-overhead methods are susceptible to noise in the dynamic environment and are not suitable for outdoor sensor platforms.

Noise-Robust Moving Object Detection

The most common approach for noise-robust moving object detection is background subtraction with a multimodal background model. In this approach, the background pixel is modeled through multiple probability distributions to cope with background objects showing dynamic motions. The most widely- used modeling method is Gaussian Mixture Model (GMM) [12], which employs a weighted sum of Gaussian distributions to describe the probability of observing the intensity at each pixel. Other multimodal background modeling approaches have been proposed based on the codebook [13], kernel density estimation (KDE) [14], and eigenspace [15]. A critical drawback of these methods, although they perform well under noise, is significant memory requirements [16], as summarized in Figure 2.1(a). Another way of identifying moving objects is to calculate optical flow (OF), the changes of motion between frames with the assumption of constant brightness. Horn-

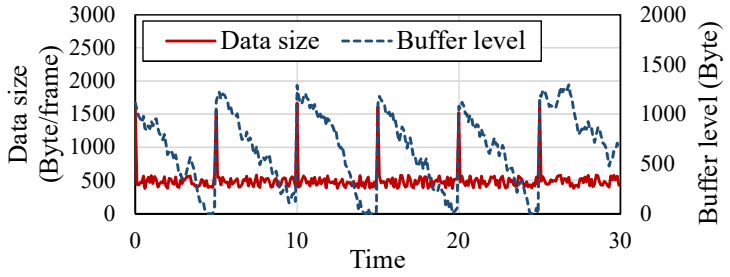


Figure 2.2: Frame data size and buffer level of the multi-rate method with a frame rate of ROI =10, non-ROI =0.2 frames/sec.

Schunck method employs low-pass filtering to reduce the noise-sensitivity [17]. Another study exploited the consistency of optical flows over a short period of time in order to cope with the salient background [18]. Although these methods are effective under dynamic background, they require substantial computation and memory resources to compute the velocity of each pixel. For example, the standard Lucas-Kanade method as implemented by Barron et al. requires 105 operations and 16 bytes of memory per pixel [19].

2.1.2 ROI-Based Coding Methods

Once the ROI is detected, a video can be processed more efficiently by focusing on the ROI. The simplest approach to ROI-based coding is to drop non-ROI blocks and encode/transmit only the ROI blocks [20]. This approach reduces encoder energy and overall data volume as non-ROI blocks are not encoded or transmitted. However, it can suffer from the loss of context information and lower overall visual quality since no visual information on the background is delivered. Moreover, in case of the false negatives, i.e., the ROI is falsely determined as the non-ROI, the ROI quality degrades significantly since no ROI information is transmitted. To address these drawbacks, Lai et al. [21] have proposed the multi-rate approach that transmits non-ROI blocks with a frame rate lower than that of ROI blocks. However, whenever the frames with non-ROI blocks are transmitted, the transmit volume increases significantly, requiring a large buffer to accommodate high fluctuation in the en-

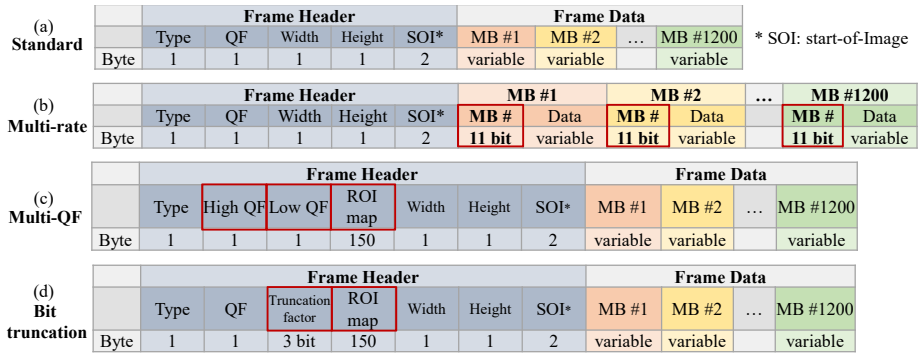


Figure 2.3: MJPEG packet structure of each approach.

coding rate [Figure 2.2]. Moreover, for correct reconstruction of the image frames without non-ROI blocks, a sensor node needs to transmit block identifiers that contain the location (or sequence number) of the blocks [Figure 2.3(b)].

An alternative (spatial) approach is to transmit both ROI and non-ROI blocks, but compress non-ROI blocks more than ROI blocks. One of the spatial approaches is to use multiple QF values in a frame; higher QF for ROI blocks, lower QF for non-ROI blocks [22]. However, the multi-QF approach also requires extra transmission of one additional QF value and the ROI map indicating whether a block is encoded using the higher or lower QF [Figure 2.3(c)]. Also, as the standard MJPEG uses a single QF in a frame, the multi-QF approach adds energy/area overhead to the MJPEG encoder/decoder to enable frame encoding with the two different QFs. Another spatial approach is to pre-process non-ROI blocks to enable higher compression at the encoder. The prior studies have proposed pre-filtering of non-ROI blocks via low-pass filters such as a median filter [23] and a Gaussian filter [24], which reduce high-frequency information in non-ROI blocks, enabling high compression with the same QF. Pre-filtering can be an attractive solution because the unit operation of median or Gaussian filtering is low-complexity. However, it is an inefficient solution for on-line tuning of non-ROI size/quality since larger filter size for more smoothening (more compression) requires heavy computation, and hence, increases computation energy [2].

2.1.3 Effect of ROI-Based Processing on Neural Network Inference

Conventionally, most of the ROI-based processing schemes are targeted to provide high perceptual quality of the ROI under the data rate constraint. These approaches assume a base station with a human operator monitoring a video scene received from the sensor node. Meanwhile, recent studies have proposed autonomous video monitoring systems to minimize human error and intervention in a monitoring process [25]. More recently, the success of deep neural networks on image classification [26][27] have introduced deep-learning based video monitoring systems in many applications such as moving object detection [28], object tracking [29], and vehicle classification [30]. In such a system configuration, the critical goal of the sensor node is to process/transmit images so that the neural networks achieve higher detection/classification accuracy. Several recent studies compared how the neural networks and human subjects perform differently on distorted images [31][32]. Another set of studies have investigated how the deep neural network performance is affected by different types of image distortions including noise [33], optical blur [34], and JPEG compression [35]. However, all the existing studies have utilized uniformly distorted images for their experiments. For a wireless sensor node with limited energy and bandwidth, allocating more resources to the ROI using ROI-based processing is necessary for better energy-quality scalability. However, no work has been done to investigate the effect of ROI-based processing on the classification performance of neural network in wireless sensor node applications.

2.1.4 Image Sensor Nodes with Energy Harvesting

In the applications of wireless image sensor nodes, the sensor nodes are usually deployed in areas where human intervention for battery replacement is a costly operation [4].Therefore, sensor nodes are expected to operate for a long period of time with limited energy sources. Longer life time can be achieved by harvesting ambient energy in the environment [36]. However, energy harvesting generally requires additional devices (thermoelectric,

Item	Tech.	Sensor integration
[4]	0.35µm	Sensor only
[38]	0.5 µm	Sensor + switch only
[39]	External sensor	Ext. Sensor + supercapacitor. Unregulated output (o/p)
[40]	0.18µm	Sensor + capacitor. Unregulated o/p
This work	0.13µm	Sensor + full DC-DC regulator with 3 regulated output

Table 2.1: System level comparison chart with prior works

piezoelectric, photovoltaic, etc.). An alternative approach to a truly self-powered sensor node design will be using the existing sensor itself as an energy harvesting device. Many wireless image sensing requires relatively low frame-rate, often limited by the channel bandwidth. Hence, the pixel array is used for sensing only for a limited fraction of time. The on-chip sensor can be configured to harvest energy during the idle time and store harvested energy in a battery or super-capacitor, providing potential of a truly self-powered system. A few recent studies have shown the feasibility of using an image pixel array for harvesting (Table 2.1) [4][37][38][39][40]. However, the existing works only considered a pixel-array and peripherals.

2.2 Data Rate Control in Image Sensor Systems

2.2.1 Target Data Rate Control

Only a limited body of work has focused on integrating ROI-based processing and the optimization of transmission energy under the variation of wireless channel conditions. To adapt system parameters to varying channel conditions, Fallah et al. proposed a link adaptation technique in which a transmitter can adaptively control its transmission parameters [41]. As the channel condition worsens, it increases transmission signal power or decreases the channel data rate to satisfy the bit-error-rate (BER) target. However, when the data rate of the channel is lower than that of the encoder, the conventional link adaptation tech-

nique, which is independent of an encoder (the independent control scheme), may result in random packet drop, leading to significant quality degradation. To address this problem, Haratcherev et al. proposed cross-layer signaling, which informs the encoder of the channel data rate so that the encoder can change its data rate as well (the feedback control scheme) [42]. However, if the feedback control scheme uses conventional rate-controlled encoders, it controls the source data rate by changing only the quality of the entire video, which may result in a low-quality ROI (the content-unaware feedback control scheme). Although the feedback controller with existing ROI-based processing approaches (i.e., the content-aware and energy-unaware feedback control scheme) can optimize the quality of the ROI, it is subject to an energy increase in the case of a channel rate decrease or a signal power increase.

2.2.2 Encoding Data Rate Control

A key challenge of applying ROI-based coding in wireless video sensing is to guarantee the data volume generated by an ROI-based encoder (encoding rate) match the available transmission data rate. However, the wireless channel bandwidth and the maximum transmission data rate can vary over time. Likewise, the encoding rate can also vary due to the variable content of video frames. If the encoder generates too much data that cannot be accommodated by the transmitter, the data should be buffered to avoid random packet drop at the transmitter. However, buffering introduces variable latency between the source and destination, and imposes energy/area overhead due to large memory requirement. Consequently, to minimize the latency and buffer requirement, there is a need for an on-line rate controller that matches the encoding rate with the transmission data rate.

The recent encoders such as H.264/AVC incorporate rate controller schemes that allocate proper bit budgets and determine a quantization parameter (QP) to minimize quality degradation based on rate-distortion-optimization [Figure 2.4(a)]. However, calculation and update of the parameters require complex computation with appreciable energy cost,

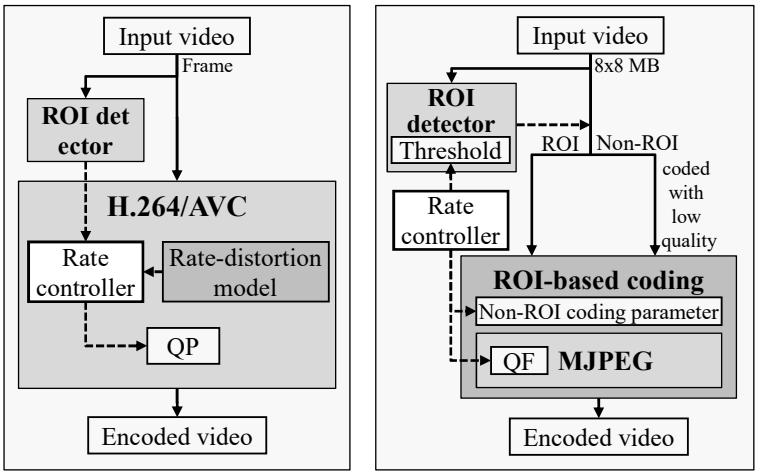


Figure 2.4: ROI-based rate controller design (a) tied to H.264/AVC and (b) based on a simple encoder (MJPEG).

and hence, it is not suitable for energy-constrained systems [43]. On the other hand, the existing ROI-based coding schemes use lower-complexity codecs such as motion JPEG (MJPEG); however, they do not employ rate controllers that can dynamically modulate the ROI-based coding parameters. Therefore, an ROI-based coding scheme with a low-complexity rate controller is necessary to improve the quality of visual information under stringent system energy constraints.

2.3 Resource-Efficient Deep Neural Networks

2.3.1 Memory Demand Reduction Techniques

There have been various approaches that compress the weights of a neural network to enhance memory-efficiency. One simple way is reducing the bit-precision of the weights to reduce both computation energy and storage requirements, as demonstrated in [44]. Instead of uniformly quantizing all the weights with the same bit precision, per-layer quantization [45][46] used different precision in each layer for higher compression ratio. However, the compression is limited by the lowest bit precision, for example, reducing precision from

32 bit to 4 bit will only result in 8X compression. To achieve higher compression, approaches have been explored to reduce the number of weights, for example, by neglecting weights (pruning connections) with magnitude below a threshold [47]. Another approach is the low-rank matrix approximation, which uses factorization to reduce the representation by two smaller matrices [48]. For lossless compression of output images of convolutional neural networks, Chen et al. used a run-length-based encoding technique [49]. There are a few works that compress the weights by transforming them into the frequency domain. Koutnk et al. has presented a method to discard high frequency components of the weights to meet the compression requirement [50]. Chen et al. has proposed to random hashing of the frequency components of convolutional neural network filters into smaller set of hash buckets [51]. Most of the preceding approaches for reducing the weight size require changing the training process to minimize the accuracy loss. For example, the approach in [52] utilizes fine tuning (retraining) after each compression stage to maintain the original accuracy. Other recent studies proposed a specialized training algorithm for binary representation of the weights [53][54]. Chung et al. also proposed a method based on matrix factorization and pruning [55], which also requires fine tuning. If the prior compression approaches are applied without modification of training algorithm (or re-training), the accuracy drop can be significant [55].

2.3.2 Computation Demand Reduction Techniques

As the computation demand of CNNs is largely dominated by convolution layers, a number of studies have explored efficient computation models for convolution layers [56]. Several methods have focused on reducing the number of convolution operations by approximating the network parameters [57]. An interesting alternative approach for fast training is to exploit the duality between spatial- and frequency-domain computation through the Fourier transforms. Recent studies [58][59] have shown the feasibility of replacing convolutions with simpler pointwise multiplications in the frequency domain (FFT-based approach).

However, this approach replaced only the operations inside the convolution layer, requiring computationally-intensive Fast Fourier Transforms (FFTs) and Inverse FFTs (IFFTs) at the boundary of every layer. Moreover, frequency-domain mapping of the parameters requires kernels to be prepared as same dimension as feature maps for pointwise multiplications, thereby significantly increasing the total memory required. In summary, although the frequency-domain approach provides fast training of CNNs, the entire network model should be carefully designed to reduce the overhead.

2.4 Collaboration of Edge and Host Platforms for DNN Inference

With recent advance in deep learning techniques, application of DNNs to the IoT environment is being actively investigated. A typical system configuration uses edge platforms for sensing visual data, which is transmitted to and processed by the host with a DNN inference engine. This configuration is appealing in the applications where the host make central decision and control, such as vehicle detection and recognition [60], remote monitoring [61], and scene analysis [62]. Although this approach relieves the inference demand for the edge platforms, its performance will largely depend on reliable transmission of the images through a wireless channel with limited bandwidth.

Recent innovation in network compression [63] and hardware/architectural acceleration techniques [64] has enabled edge platforms with an integrated image sensor and deep learning inference engine [65]. When these edge platforms are used to perform the entire inference for delivering the result to the host, large resource demand of deep neural networks will limit their performance.

As opposed to the entire inference at the edge or at the host, a recent study presented a distributed structure of edge devices performing inference of a shallow part of the network [66]. However, network partitioning presented in this study was not based on the energy and throughput analysis of the system. Moreover, it did not apply an encoding technique to the intermediate-layer features before transmitting them. Although a few studies have

investigated the accuracy impact of input image encoding [67] and the weight compression [63], there has been no work that explored the effect of intermediate-layer feature encoding on the neural network performance.

CHAPTER 3

RESOURCE-AWARE IMAGE SENSOR SYSTEM DESIGN

3.1 Design of a Low-Power Image Sensor System

3.1.1 Introduction

Wireless video sensor nodes in the IoT environment are usually powered by limited energy sources. Therefore, the sensor system design should be optimized under a stringent system energy constraint [68]. Since a major source of its energy consumption is wireless transmission, an efficient video compression algorithm should be used to reduce the amount of data being transmitted to the wireless network [69]. Although most conventional compression algorithms assume equal importance of every region in a video, users pay more attention to their regions of interest (ROI) in many video applications [3]. For example, in a remote surveillance application with a fixed camera, regions with moving objects are more important than the background. Therefore, by acknowledging the relative importance of an ROI, we can enhance the energy efficiency of a system while preserving the quality of an ROI. However, ROI-based processing can be computationally expensive, for example, hardware engines for accurate motion estimation as used in H.264 standards require significant energy and area [70]. Therefore, to design a ROI-based video-processing approach suitable for resource constrained sensor nodes, we must investigate a system-level tradeoff between energy and the ROI quality. In this section, I propose an energy-efficient wireless video sensor node with content-aware pre-processing and an energy- and content-aware feedback control scheme. This work proposes content-aware pre-processing, which reduces both computation and transmission energy by selectively encoding and transmitting regions with moving objects in a video. For energy- and area-efficient detection of moving objects, I design a simple pre-processor based on edge detection and frame differ-

encing, which provides an approximation of moving object detection. The content-aware pre-processing is combined with the standard MJPEG encoder, improving both rate-quality and energy-quality trade-offs over the conventional MJPEG and H.264-intra based encoding approaches. I further reduce the area and energy of the proposed system through block-level pipelining and an optimal voltage/frequency assignment with power gating. The full-chip design and synthesis in the 45nm predictive technology model (PTM) library show that the pre-processor adds only 3.0% energy and 5.5% area overheads to the standard MJPEG. Analysis with a color video of 160 x 120 pixel resolution (twelve-bit pixel depth) shows that the sensor node can operate within 1.1 uJ/frame computation energy at a frame rate of ten frames/second.

3.1.2 Key Design Concepts

System Design

The proposed system aims to maximize the quality of the ROI under varying channel conditions with less area and energy consumption. This goal is achieved by the following design concepts: 1) simple pre-processing for moving object detection, 2) an energy- and content-aware control scheme for adapting to channel conditions, and 3) low-power design techniques. The block diagram of the proposed system is shown in Figure 3.1. First, the pre-processor calculates the temporal activity of edges for each macroblock (MB). Since the standard MJPEG encoder processes the image with the unit of an 8x8 pixel block, the MB size other than 8x8 pixels requires additional buffer and control logic. Therefore, I use an 8x8 pixel MB for pre-processing in this work, but varying MB size would be interesting future work. After each MBs activity is calculated, MBs with activity levels higher than the pre-defined threshold are encoded by the MJPEG and transmitted, while MBs with lower activity levels than the threshold are dropped. As the threshold can change the number of MBs to process in a frame, it can be used as a control parameter to regulate the source data rate together with the quality factor (QF) in the baseline MJPEG encoder. The

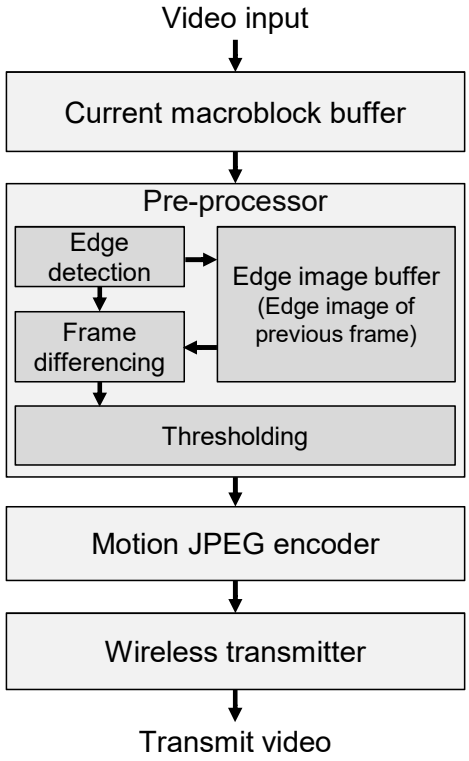


Figure 3.1: Block diagram of the proposed system.

source data rate is determined so that the target transmission energy is maintained even in a variation of a wireless channel. Each module employs low-power design techniques including block level pipe-lining and power gating. I use the MJPEG encoder in this work because it is more efficient in terms of area and energy consumption than recent encoders such as H.264 or MPEG. These encoders are computationally complex because they use motion compensation techniques to remove temporal redundancy for higher compression ratio [71]. Meanwhile, the baseline MJPEG does not exploit dependency between frames, resulting in lower en-coding effectiveness. Therefore, to improve encoding effectiveness with a small increase in complexity, I employ a simple pre-processor that eliminates temporal redundancy (stationary background objects) by examining the dependency between two consecutive frames.

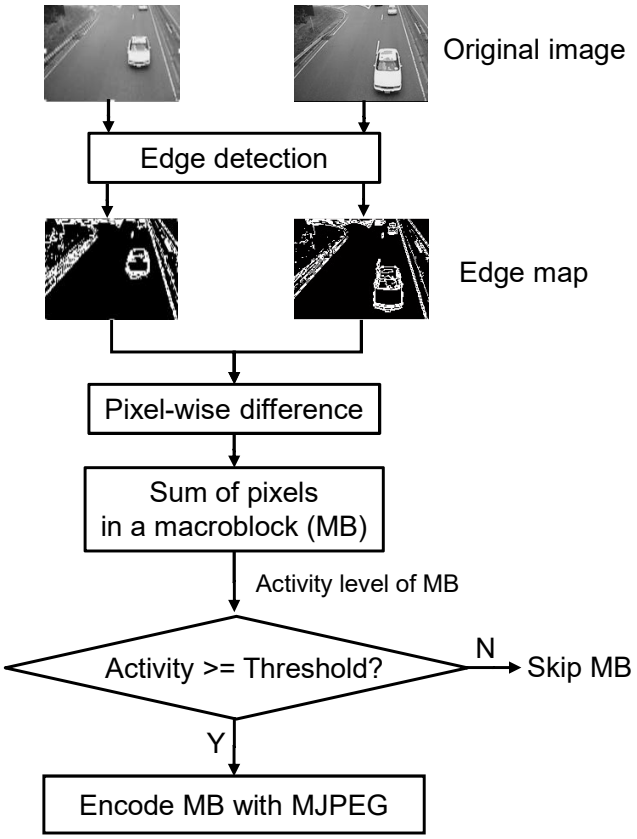


Figure 3.2: Procedure of the content-aware pre-processing.

Content-Aware Pre-processing

In order to perform moving object detection in a low-power sensor node, the pre-processor should adopt an energy- and area-efficient algorithm. At the same time, the algorithm should correctly detect moving objects to maximize both the reduction in the transmission energy and the quality of the ROI. As ED is efficient in memory requirement and boundary detection and FD can remove temporal redundancy, I employ a sequential combination of ED and FD to take advantage of the two methods. The procedure of the proposed algorithm is shown in Figure 3.2. After performing edge detection using the Sobel operator on the current frame n, we obtain an edge map E_n in which the pixel value is 1 on an edge and 0 otherwise. Then, we calculate the absolute pixel-by-pixel difference between E_n with $E_{(n-1)}$ and sum them up for an MB to obtain the activity level of i-th MB in a frame n, A_n^i,

i.e.,

$$A_n^i = \sum_{(x,y)MB_n^i} |E_n(x,y) - E_{(n-1)}(x,y)|. \tag{3.1}$$

If the activity level of an MB is larger than or equal to the threshold, the MB is sent to the encoder. Otherwise, the MB is not processed. That is, MBs that have more pixels with varying edges than the threshold are marked as ROIs. In a color video with a YUV format, the Y component, which represents the luminance of the color, is a good candidate for edge detection [72]. Therefore, to reduce the computation, pre-processing is performed only on the Y component. Since the U and V components are normally in half-resolution of the Y components, the number of MBs in the Cb or the Cr components is less than that in the Y component. Therefore, if at least one MB in the Y component is determined as the ROI, the corresponding MBs in the Cb and the Cr components are also marked as the ROI to correctly represent the color of ROI MBs. The most important advantage of the proposed method is its high efficiency in terms of energy and area. Since it is based on computationally inexpensive operations such as edge detection, frame differencing, and thresholding, it is of low complexity. It is also area-efficient because it requires only one-bit memory per pixel to store the previous edge map, while FD necessitates eight-bit memory per pixel to store the original data of the previous frame. In addition to the high efficiency, the proposed approach also yields a high performance in detecting moving objects. It is robust to dynamic scene changes since the edge information on the objects remains significant even in a variable environment. Moreover, it reduces false detection on stationary objects since the FD operation followed by ED removes temporal redundancy on the edge locations of the stationary objects over two consecutive frames. Even if slight temporal changes in their edge locations generate noise in the background MBs, the thresholding operation followed by ED and FD correctly drops these MBs since their activity levels are usually lower than that of ROI MBs.

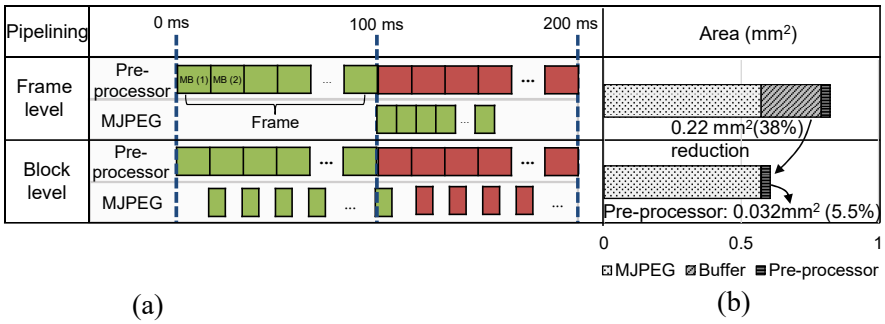


Figure 3.3: Effect of pipelining level. (a) Diagram. (b) Area estimation.

3.1.3 Low-Power Encoder Design

The MJPEG encoder with pre-processing is designed in the 45-nm PTM library. To increase energy efficiency of the system, I employ low power design techniques including block-level pipelining and pow-er gating.

Pipelining

The proposed system has two pipeline stages (the pre-processor and the MJPEG), in which we can apply two kinds of pipelining schemes: block- and frame-level pipelining, shown in Figure 3.3(a). In block-level pipelining, once the pre-processor processes one MB, the MB stored in a buffer is fetched by the MJPEG encoder. By contrast, frame-level pipelining stores and processes the video frame by frame. Our design uses block-level pipelining since it allows lower latency and a smaller buffer. This advantage de-pends on the size of an input video. If we assume a video with 160 x 120 pixels and 300 MBs in one frame, the latency and required buffer size for block-level pipelining is 300X lower than that required for frame-level pipelining. Figure 3.3(b) shows that block-level pipelining reduces the area by 38% compared to frame-level pipelining.

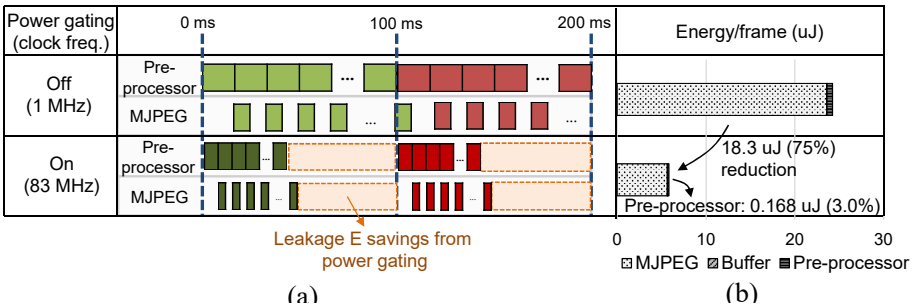


Figure 3.4: Effect of power gating. (a) Diagram. (b) Energy estimation.

Power gating

Additional energy savings can be achieved by using power gating coupled with near-threshold operation (VDD=0.5V) since it reduces leakage energy that is proportional to the on-time of the design, while dynamic energy is independent of an operating frequency. As the on-time decreases with a higher frequency, we can save significant leakage energy by shutting down the design after processing the current frame with an operating frequency higher than the minimum frequency, which is determined by a frame rate [Figure 3.4(a)]. Assume a frame rate of ten frames per second, which requires the operating frequency of 1MHz to satisfy the timing. If the system operates at the maximum operating frequency (83 MHz @0.5V), the encoder processes one frame within 1% of the frame interval. As Figure 3.4(b) shows, power gating reduces energy consumption by 75% when we assume no MB is dropped at the pre-processor. It should be noted that the energy saving due to power gating will be larger when the pre-processor drops non-ROI MBs, which is the more likely scenario.

		ED	FD	ED+FD
Area (um^2)	Logic	1,923	2,295	9,533
	Buffer	0	219,000	22,000
	Total	**1,923**	**221,295**	**31,533**
Energy per fra me (nJ)	Logic	28	7	148
	Buffer	0	288	20
	Total	**28**	**295**	**168**

Table 3.1: Area and energy of moving object detection methods.

3.1.4 Experimental Results

Experimental Framework

For system-level performance analysis, I use a Nordic Semiconductor nRF24L01+ transmitter [73]. I compare the performance of the proposed system with the baseline MJPEG and the intra-coded H.264/AVC, which is the standard H.264/AVC us-ing intra-frame prediction only. Their hardware implementations are synthesized using the Synopsys Design Compiler, and the area and energy of the synthesized implementations are measured by the Syn-opsys Primetime tool. The quality of an encoded video is measured by software models; for the baseline MJPEG and the proposed system, I develop a MATLAB/Simulink-based model, and for H.264/AVC, I use the JM 18.6 reference SW [74]. I use four video sequences: traffic and atrium, provided by Simulink tool, and hall monitor and bridge (close). All the sequences are 160x120 pixels and YUV 420 format with ten frames per second. As a quality metric, I use the structural similarity (SSIM) index for color video [75] of ROI MBs.

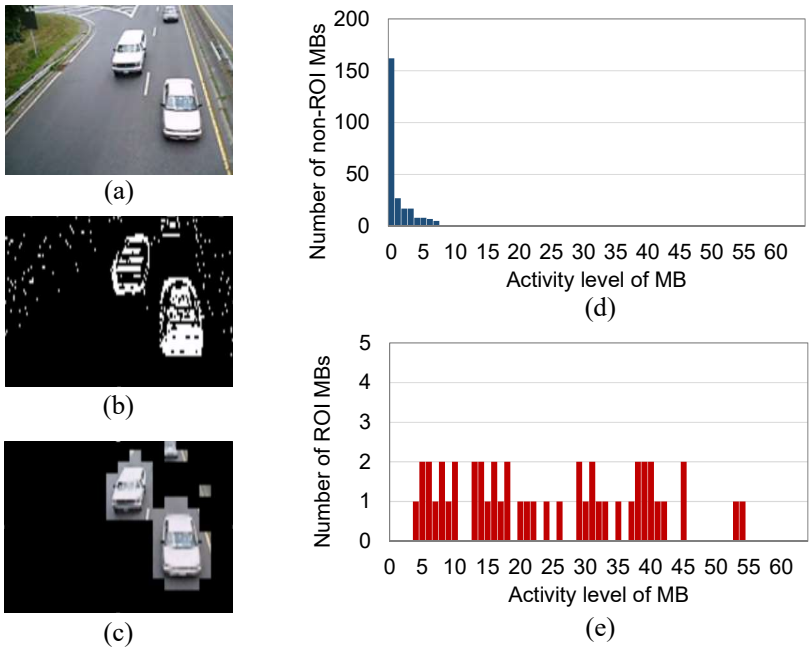


Figure 3.5: Results of the proposed pre-processing operation. (a) Original frame image. (b) Frame image after ED and FD (difference of the two consecutive edge maps). (c) Frame image after thresholding. (d) Distribution of the activity levels of non-ROI MBs. (e) Distribution of the activity levels of ROI MBs.

Experimental Results

In order to analyze moving object detection performance of the proposed pre-processor, I examine the results of a combination of ED and FD operation. As shown in Figure 3.5(b), if the edge locations of the background objects change in the successive frames, the inter-frame difference of edges suffers from noise, which may result in the false detection on the back-ground. However, as the noise is scattered, its sum in an MB (the activity level) is relatively low compared to the activity levels of ROI MBs as shown in Figure 3.5(d) and (e). Therefore, by using the thresholding operation after ED and FD, we can exclude MBs with a stationary background as shown in Figure 3.5(c). In summary, although the activity level calculated by a combination of ED and FD may indicate activity on the background, a simple thresholding operation can reduce the false detection on the background objects. One way to further reduce the false detection would be applying more complex object

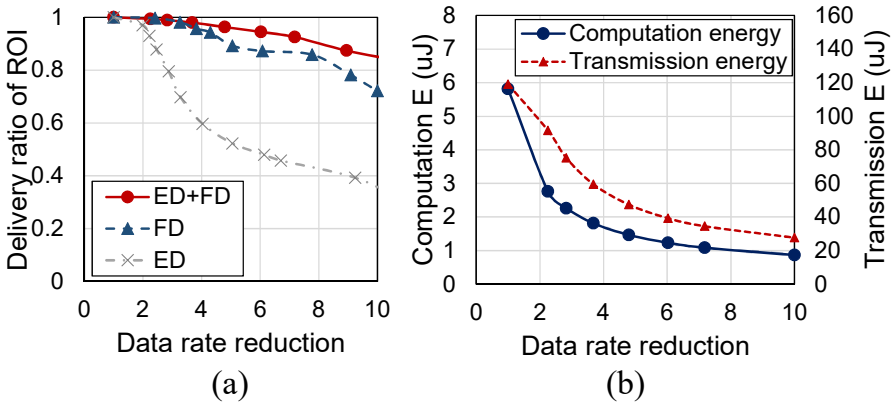


Figure 3.6: (a) Performance comparison of moving object detection methods for traffic video. (b) Computation and transmission energy consumption using proposed pre-processing (ED+FD).

recognition techniques, which can be interesting future work.

Figure 3.6(a) and (b) show the performance of the moving object detection schemes measured in terms of how many ROI MBs are correctly identified (delivery ratio of the ROI). Table 3.1 shows their area and energy consumption. While ED is most energy efficient, it shows the worst detection performance. Although ED+FD has a higher logic area and energy than FD, the total area and energy is smaller since it requires a smaller buffer for an edge image of the previous frame (1 bit/pixel), while FD requires the original previous frame (8 bit/pixel). The proposed pre-processor (ED+FD) incurs only 3.0% energy and 5.5% area overhead to the baseline MJPEG [Figure 3.3(b) and 3.4(b)]. At the system level, we can save 79% of computational energy and 67% of transmission energy while ensuring the delivery of 95% of ROI MBs [Figure 3.6(b)].

To evaluate the performance of the content-aware control scheme, I compare its rate-quality characteristics with content-unaware compression schemes using four video sequences. For all the sequences, the proposed system yields better quality at the same amount of data reduction as shown in Figure 3.7. For ex-ample, with the atrium video, the proposed system shows the SSIM of 0.8 with 100X data reduction, while the SSIM of

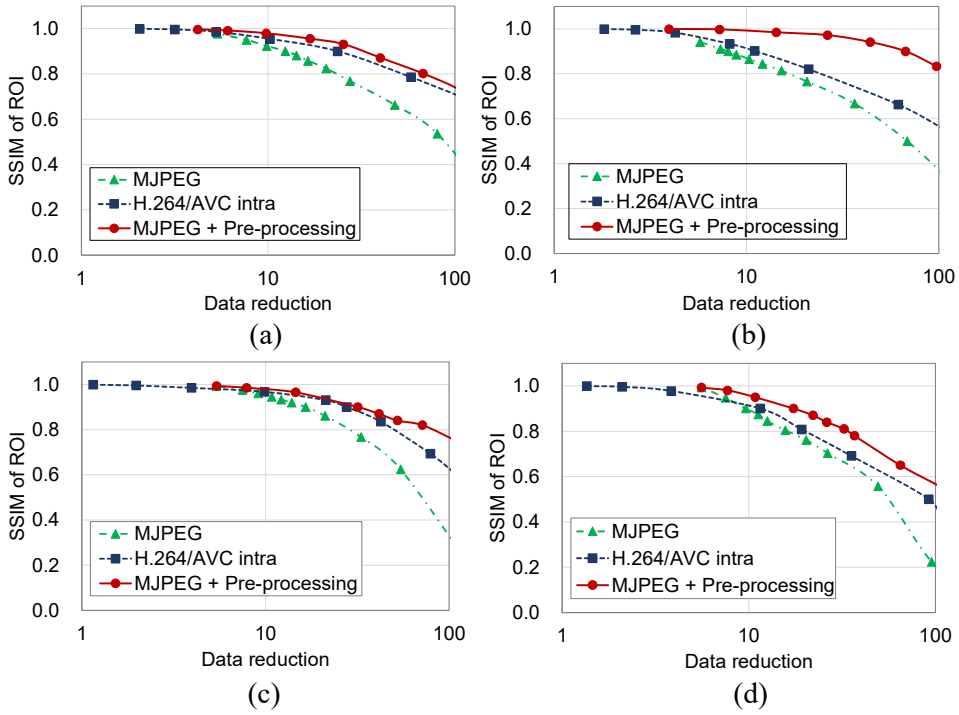


Figure 3.7: Rate-quality characteristics of each compression method. (a) traffic video. (b) atrium video. (c) Hall monitor video. (d) bridge video.

the MJPEG and the H.264/AVC fall below 0.6 and 0.4, respectively.

Our design shows better rate-quality characteristics as it drops mainly non-ROI MBs and encode ROI MBs in a higher quality. As Figure 3.8(a) shows, control-ling the threshold with a fixed QF provides better rate-quality performance than controlling the QF without thresholding (content-unaware control). However, if the data reduction using thresholding exceeds certain point, the quality drops significantly since it starts to drop ROI MBs. Rather than dropping ROI MBs with a high threshold, we can switch to a lower QF to yield better quality at the same amount of data reduction. Therefore, the data reduction using both the threshold and the QF provides better rate-quality tradeoff. Also, an input video with fewer ROI MBs (atrium video) has better rate-quality characteristics than a video with more ROI MBs (traffic video) [Figure 3.8(b)] since more non-ROI MBs can be dropped in a video with fewer ROI MBs.

Figure 3.9 shows that our design also offers better energy-quality performance than

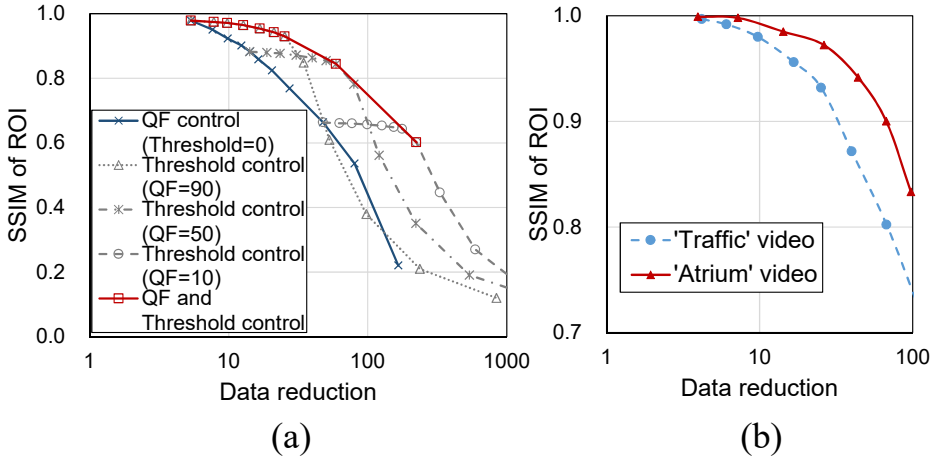


Figure 3.8: (a) Rate-quality characteristics of QF-only, threshold-only, and both QF and threshold control. (b) Rate-quality characteristics of the proposed control for traffic and atrium video.

content-unaware compression methods. For the SSIM of the ROI=0.9 in the traffic video, it consumes 56% and 23% less energy than the baseline MJPEG and the intra-coded H.264/AVC, respectively, as shown in Figure 3.9(a). It should be noted that our design consumes less energy in both computation and transmission [Figure 3.9(e)].

3.1.5 Summary of the Section

This section presented an encoding engine with low-overhead pre-processing that enables energy- and content-aware data rate control for reliable and energy-efficient transmission of information in wireless video sensor nodes. The PID-based rate controller is presented to dynamically tune algorithmic parameters to yield higher quality of the ROI while maintaining the target energy consumption under varying channel conditions. The proposed design achieves low-power consumption as well as tolerance to variation in environmental conditions such as an input video stream or a wireless channel; both are vital requirements in the design of the resource-constrained sensor nodes for IoT. The proposed low-power pre-processing and bounded-energy operation can enable sensor nodes in the IoT to per-

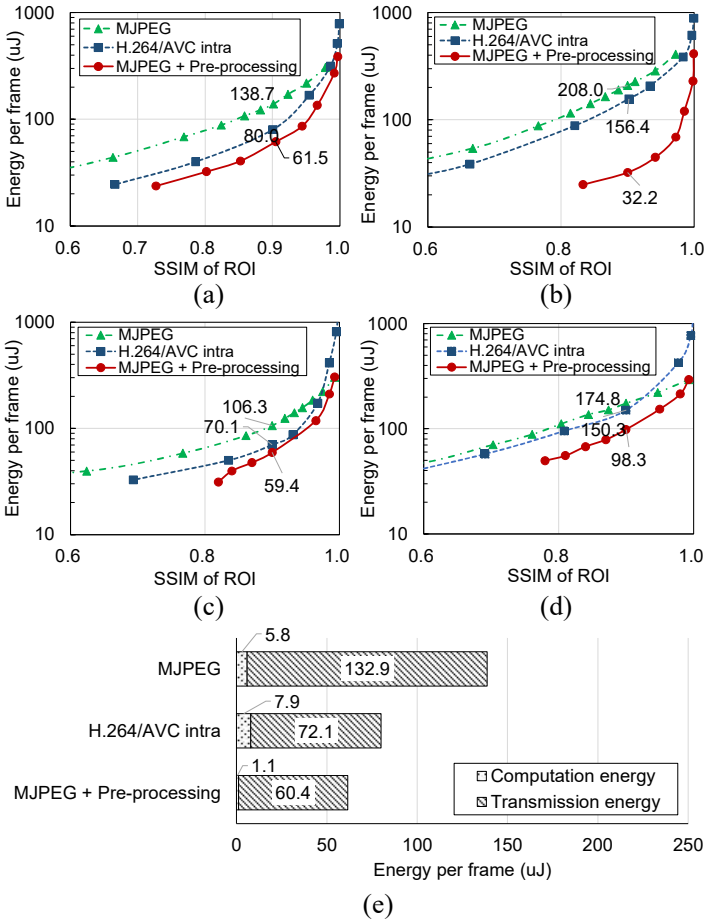


Figure 3.9: Energy-quality characteristics of each compression method for (a) traffic video, (b) atrium video, (c) Hall monitor video, and (d) bridge video. (e) Energy consumption breakdown for each encoding schemes.

form more operations with energy constraints. Since wireless communication of the IoT is unreliable by nature, it is also important to adapt to wireless channel variation for reliable delivery of information. The proposed dynamic source-rate controller combined with the system-wide feedback control scheme preserves the quality of the ROI under variations in channel conditions as well as input video characteristics. Therefore, the proposed design can be a lightweight alternative to complex video encoders for resource-constrained sensors.

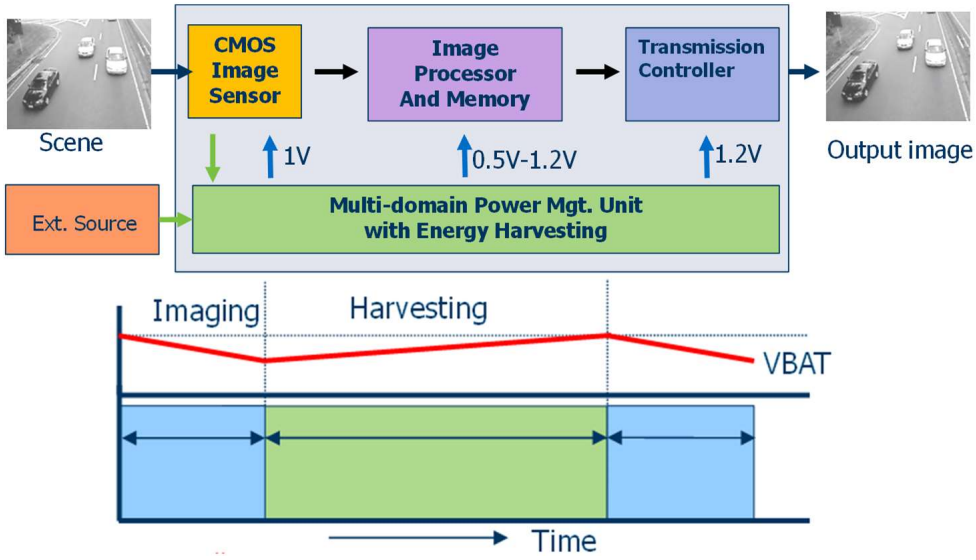


Figure 3.10: System architecture of the image sensor node that can be operated by energy harvesting from the CMOS image sensor.

3.2 Implementation of a Self-Powered Image Sensor System

3.2.1 Introduction

In the previous section, I presented a design of a low-power wireless image sensor node that can be applied in many areas such as traffic monitoring, remote sensing, and military surveillance. In these applications, sensor nodes are usually deployed in areas where human intervention for battery replacement is a costly operation [4]. Therefore, sensor nodes are expected to operate for a long period of time with limited energy sources. Longer life time can be achieved by harvesting ambient energy in the environment [36]. However, energy harvesting generally requires additional devices (thermoelectric, piezoelectric, photovoltaic, etc.). An alternative approach to a truly self-powered sensor node design will be using the existing sensor itself as an energy harvesting device. Many wireless image sensing requires relatively low frame-rate, often limited by the channel bandwidth. Hence, the pixel array is used for sensing only for a limited fraction of time. The on-chip sensor

can be configured to harvest energy during the idle time and store harvested energy in a battery or super-capacitor, providing potential of a truly self-powered system. A few recent studies have shown the feasibility of using an image pixel array for harvesting. However, the existing works only considered a pixel-array and peripherals.

As an important goal of a sensor node is to deliver the visual information, it is generally integrated with processing and communication engines. Therefore, a complete image sensor generally includes an image processor, memory, a power management unit, and a transmission controller. Integration of an image sensor system can be performed in a board level by wiring separate component chips. However, this approach increases the form factor of the system, and inter-chip connections between the components can reduce energy-efficiency of the system. A system-on-chip (SoC) integration of all the components in a single chip can make the system more compact. Also, it allows the optimization of the data path and control logic by considering interactions between the components. An integrated sensor system that can be self-powered by energy harvesting has not been studied yet. In this section, I present a single-chip system to study the feasibility of a self-powered image sensor node with integrated sensing, energy harvesting, power management, and processing [Figure 3.10]. The mixed-signal SoC integrates an energy harvesting pixel array with multi-output power management unit, a digital processing engine with an on-chip SRAM for moving object detection, and a transmission controller. The innovation of this work lies in the system design and analyzing the effects of component-level design choices on the self-power performance of the sensor. The system incorporates a reconfigurable CMOS Active Pixel Sensor (APS) array that was presented in [76]. The pixel-array operates as a photo detector while sensing/processing a frame and as a photovoltaic cell in between successive frames to harvest energy. The CMOS sensor is coupled with an on-chip power management unit (PMU) that has been published in [77]. The PMU multiplexes a single inductor for boosting the harvested input for intermediate energy storage, as well as generating three regulated output voltage domains. An autonomous mode management unit

(AMM) controls the switching between imaging and harvesting mode of the sensor and transitions between the boost and buck mode of the regulators. The mode switching can be externally controlled or autonomously managed based on available stored energy. An image processing engine with an on-chip SRAM identifies the region with moving objects to reduce transmission energy with reliable delivery of information. The system is designed to enhance self-power performance by reducing both processing (active mode) power demand and standby energy dissipation in between frames. As the SRAM needs to remain powered-on between frames, reducing the memory size, and hence, total leakage, is a key consideration in the proposed design. The following design choices are explored to enhance self-power performance: A lightweight algorithm is used for moving object detection with reduced memory and computation, and hence, lower power demand. This algorithm eliminates the need for frame buffer thereby significantly reducing the on-chip memory size. To reduce energy of the system, I minimize the size of the buffer between the image sensor, processor, and transmission controller. Therefore, instead of frame-level pipelining, a block-level pipelining scheme is applied to have only block-sized buffers between the components. The block-level pipelining is enabled by a block-wise readout at the CMOS sensor instead of the conventional row-wise readout. The sensor peripheral is designed to support the block-wise readout. The processing engine operates at highest voltage and frequency, instead of low-voltage operation, to enhance the idle periods between frames, and hence, harvested energy. The digital logic is powered down in between frames by controlling the PMU output to reduce leakage energy (frame-level power-gating)

A test-chip is designed in a 0.13m CMOS technology node. It can process image frames with 128x96 pixels at the maximum frame rate of 230 frames/sec. The image processor performs the expected function of moving object detection. The design demonstrates the peak harvested power of 2.1W at the output of the sensor array. Based on the peak harvested power and the measured power dissipation of the different components, I estimate the sensor can be self-powered while processing a frame every 15 seconds. Further,

adaptive supply voltage control of the SRAM can enhance the minimum frame interval for self-powered operation to 7 seconds/frame.

I also analyze the challenges and opportunities to enhance the self-power performance by studying interactions of different components of the system. The simulation-based analysis is presented to understand the effect of increasing pixel size, design modification of the PMU for higher efficiency of the boost converter, and use of low-power RF transceivers on self-power performance. The measurement shows the captured image has high level of noise that not only degrades the image quality, but also causes false detection of the ROIs, thereby increasing volume of transmitted data. I perform sensor noise analysis and propose algorithm/hardware of a noise-robust moving object detection scheme for better image quality, reduction of transmitted data volume, and improved self-power performance.

3.2.2 System Design

System Design

A block diagram of the proposed system is shown in Figure 3.11. The CMOS image sensor captures an image in a unit of an 8x8 pixel macroblock (MB) of a frame, and stores it in a MB buffer. Then, the image processor determines whether an MB contains moving objects (Region-of-Interest, ROI) or not. Only the ROI MBs are transmitted, while non-ROI MBs are dropped. Rather than the conventional row-wise readout, the CMOS sensor performs a block-wise readout, which enables block-level pipelining between the CMOS sensor and the image processor. Block-level pipelining reduces the latency and overhead from a buffer between the CMOS sensor and image processor, since it does not require the system to wait for the entire frame to be read and stored [Figure 3.12(a)]. Also, I enhance the energy efficiency by applying a frame-level power-gating technique and higher operating frequency. This is due to the increased idle period, during which the CMOS sensor can harvest energy and the digital components are powered down to reduce leakage energy [Figure 3.12(b)]. The system is powered by an energy storage element (battery or high-density capacitor),

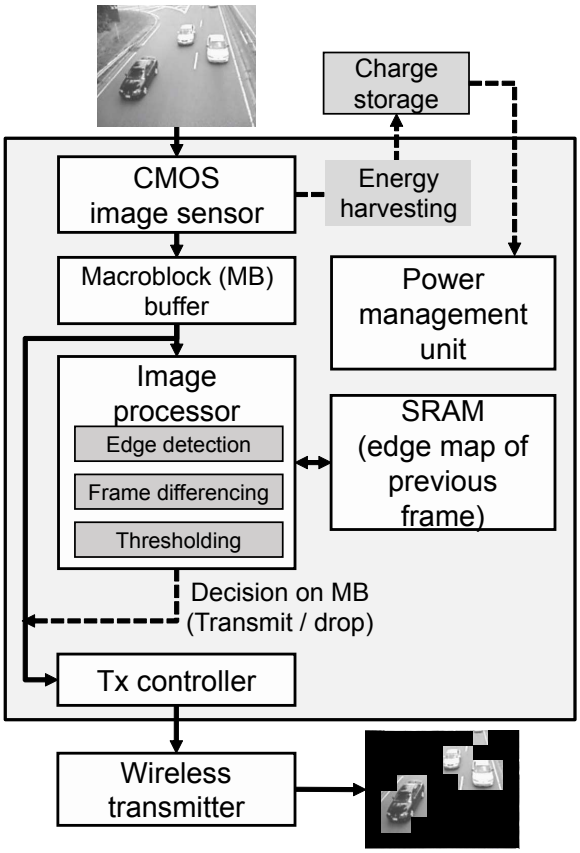


Figure 3.11: Block diagram of the image sensor system

which can be charged by the energy harvested from the reconfigurable 128 x 96 CMOS Active Pixel Sensor (APS). The APS array operates in a photoconductive mode when imaging and a photovoltaic mode when harvesting. The mode switching can be based on the frame rate, and/or autonomously managed based on available stored energy. The power management is performed using a single inductor boost/buck regulator. The boost regulator harvest energy from the sensor to charge on-board storage using maximum power point tracking. The stored energy is then delivered to three independent load domains using a Single Inductor Multiple Output (SIMO) buck regulator. The boost and SIMO buck shares a single inductor using an autonomous on-chip time multiplexing controller. The system is designed to have off-chip charge storage and wireless transmitter to provide a flexible and compact single-chip solution.

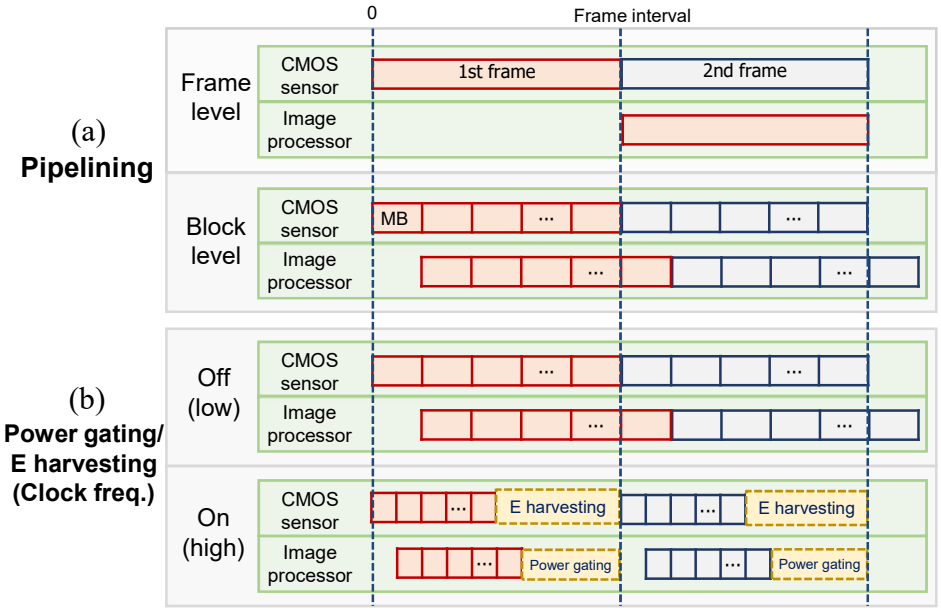


Figure 3.12: The effects of the low-power design techniques: (a) pipelining and (b) power gating

Autonomous Mode Management (AMM)

The AMM controls the switching between imaging and harvesting mode of the sensor (depending on frame rate and energy availability) as well as transitions between the boost and buck mode of the regulators (depending on the priority of the load and battery voltage level). In applications with a target frame rate, a frame rate control signal will cause EH to change from low (sensing) to high (harvesting). A low EH forces buck only operation to enable sensing, processing, and communication of a frame. The high EH on the other hand, enables both boost and buck operation between sensing two frames. The boost mode is necessary to harvest energy and replenish the energy storage. The buck operation remains necessary for some circuit blocks even between frames. For example, preserving the previous frame data is often required for compressing image with temporal redundancy, and hence, on-chip memory should be powered in between two frame capturing [38]. The AMM enforces (i) boost-only operation when battery voltage (V_{BAT}) is less than a lower

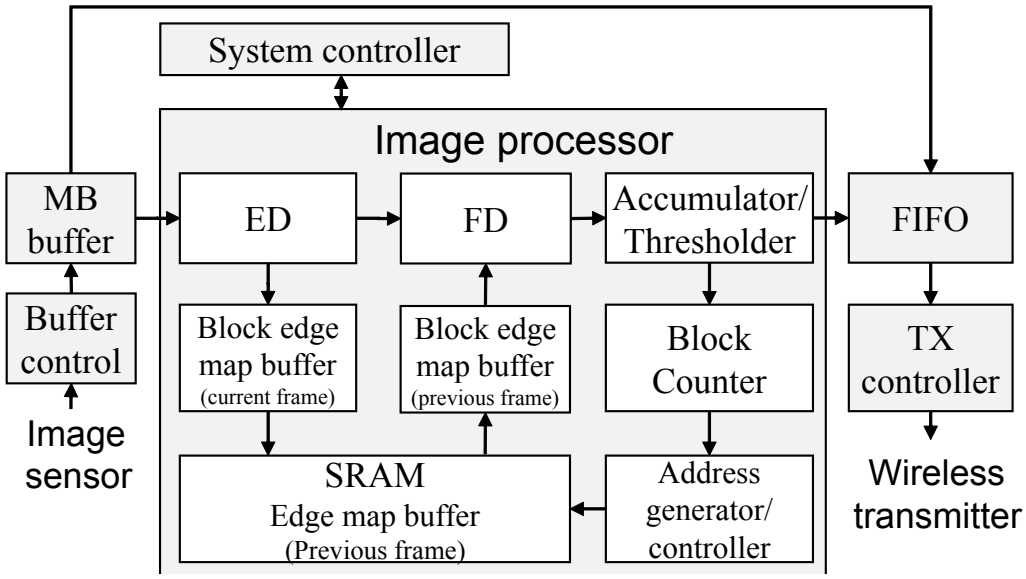


Figure 3.13: Block diagram of the digital units (image processor, transmission controller).

limit (LL) (deep discharge cut off point), (ii) buck-only operation if V_{BAT} is higher than a higher limit (HL) to prevent overcharging, and (iii) switching between buck and boost modes when: $LL \leq \mathrm{V}_{BAT} \leq$HL.

In energy autonomous imaging mode, the EH is self-generated by the system. Such decision is made by sensing the voltage drop in the energy storage and assessing how much energy is required to process the next frame. If the energy level in the storage is below that minimum limit, the system decides to harvest before allowing next frame capturing. Thus, in the self-powered case, the frame rate becomes a system defined variable and varies depending on available energy. In practical operation, the demanded frame rate can push the system into sensing but if enough energy is not available, the AMM will stop sensing and will go to harvesting mode.

Image Processor

A block diagram of the digital units (image processor and transmission controller) is shown in Figure 3.13. The image processor performs moving object detection to reduce the transmission energy by dropping the static background region. As an underlying moving ob-

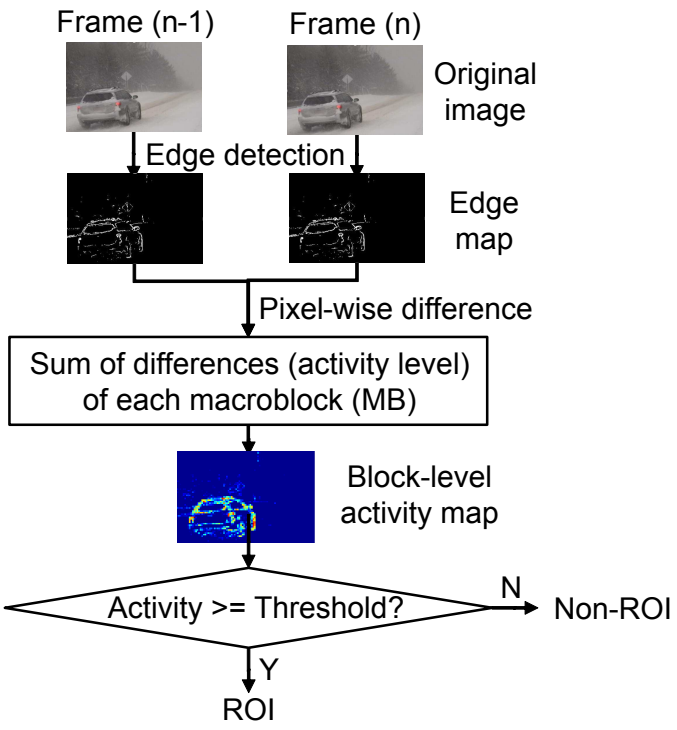


Figure 3.14: Data flow of the moving object detection method based on edge detection and frame differencing.

ject detection method, I use the approach based on edge detection and frame differencing (ED+FD) [11]. As Figure 3.14 shows, each MB in a frame is processed by edge detection to create an edge map. Then, I calculate the sum of absolute pixel-wise difference between two consecutive edge maps, which we call the activity level of an MB. If an MB has the higher activity level than the threshold, the MB is determined as ROI. Only the ROI MBs are transmitted, while non-ROI MBs are dropped. Therefore, the threshold can be used as a knob to control the number of MBs to transmit. The threshold value can be configured from zero to 14 with a step of two through a 3-bit input signal. As described in Section II.B, the image pixel is designed based on the logarithmic structure, which enables logarithmic compression that allows image sensing over a wide range of light levels. The logarithmic compression has the effect that low intensity pixel values are enhanced at the expense of contrast sensitivity reduction in the high pixel values. Therefore, the edge detection performance can vary depending on the light intensity of the objects. For example, edges

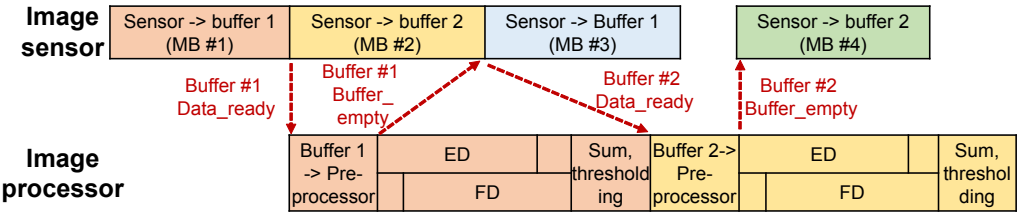
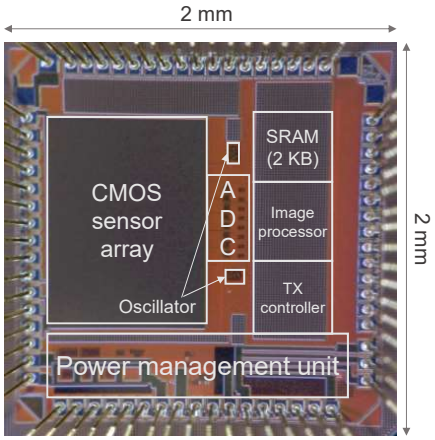


Figure 3.15: Timing diagram of the pipelined process between the image sensor and image processor.

detection of objects in high light intensity can become difficult due to reduced contrast. To deal with this problem, the system has 3-bit input signal that determines the edge detection threshold, which is used to control the detectability of edges (different from the activity threshold used to control the ROI detectability). When there is low contrast between the objects, we can set the threshold low to reduce undetected edges.

When the non-ROI MBs are dropped by the image processor, only part of the frame will be delivered to the receiver node. For correct reconstruction of the frame at the receiver, the information on the MB location should be transmitted together with the image data. Therefore, we attach a header with an 8-bit block number at the beginning of each MB data. For flow control of block image data from the image sensor, I use a handshaking protocol with two 64-byte buffers. Figure 3.15 shows a timing diagram of the pipelined processing of the image sensor and image processor. First, the image sensor writes one MB into the MB buffer #1. After it finishes writing, it raises $DATA\_READY$ signal, which initiates moving object detection process (ED+FD) in the image processor. While the image processor reads data from the buffer #1, the image sensor writes the next block data into the buffer #2. To minimize the latency, the image sensor is designed to start writing the block data when there is at least one empty buffer. After the image processor completes reading, it raises $BUFFER\_EMPTY$ signal, which allows the image sensor to write the next block data to the buffer. If this signal is low when the image sensor finished writing the block, it waits until the signal goes high. Conceptually, the ED+FD method first creates an edge map of an MB, then performs frame differencing with the corresponding MB in

the previous frame. However, when ED+FD is processed serially, the total latency of the ED+FD process will be the sum of each ED and FD process. To reduce the latency, I apply a kernel-level pipelining between the ED and FD processes. As Fig 3.15 shows, once 3x3 edge detection kernel generates a 3x3 portion of the block edge map, it is frame-differenced with the corresponding area in the previous frame. As explained above, I utilize various design techniques to minimize the frame processing latency. Also, under a given frame rate constraint, reduced latency will increase the frame idle time, which leads to the increase in harvested energy. In an autonomous mode, lower frame processing latency will decrease the total frame interval, enhancing the system self-power performance. Our moving object detection method is energy- and area- efficient since it is based on simple edge detection and frame differencing operations. More importantly, it is memory-efficient since it requires only 1 bit/pixel to store the edge map of the previous frame. As the sensor array has 128x96 pixels, the system requires 1.5kB of memory. The SRAM consists of two instances of a 1kB macro, making the total system capacity 2kB. Each 1kB macro is further subdivided into four 256B sub-banks, with each sub-bank containing 32X64 SRAM bitcells. The area for a 1kB macro is 0.0775 mm^2. The data bus width for each macro is 32 bits, with separate ports for read and write, and enable signals are used by the image processor to determine which macro will be written to. In order to reduce power consumption and design complexity, the SRAM bitlines are designed to operate with rail-to-rail swing, and utilizes single-ended sensing which eliminates the use of power hungry differential sense amplifiers. The leakage power in idle state was measured to be 1W for 2kB SRAM. A pow er-gating controller determines when to power down the rest of the image processor. As mentioned earlier, the logic engine of the image processor, except for the SRAM, is power-gated to avoid leakage energy consumption during the frame idle period. The power-gating signal (PG) should be delivered to the power management unit to disable the power delivery to the image processor. The PG signal is generated by the always-on power-gating controller in the image processor, and goes high when the last block of the frame is transmitted through the



(a)

Specification	Value	
Technology	IBM 0.13µm	
Area	2 x 2 mm^2	
Supply voltage	1.2 V	
Max. frame rate	230 frames/sec	
Energy consumption per frame (µJ, @230fps)	Sensor array	0.07
	ADC, peripherals (from simulation)	5.7
	Image processor	3.0
	Tx Controller	4.5
	SRAM	0.24
Energy harvesting	2.1µW @200klx	

(b)

Item	Value
No. of domains	4
Cold start	370mV
Max. VBAT/VOUT range	3.3V/0.2V -1.2 V
Bias power	0.2µW
Max. efficiency (Boost/Buck)	76%/54%
Max. power	10mW
Area(mm^2)/ Technology	0.6/ 130nm
Battery mgt.	Yes
Cross reg mgt.	Yes

(c)

Figure 3.16: (a) Die photo of the sensor node and key performance parameters of (b) the system and (c) the PMU.

transmission controller. It remains high until the EH goes low, which initiates the beginning of the next frame processing.

Transmission Controller

The transmission controller is designed for data transmission through a Nordic Semiconductor nRF24L01+ wireless transmitter. To deal with the mismatch of the data rates from image processor and a transmitter, the controller includes a FIFO with the size of two 32-byte payload. When the FIFO is full, the transmission controller sends a signal to the image processor to stop pushing the data. When received the signal the image processor is clock-gated until all the data in the FIFO is transmitted.

3.2.3 Measurement Results

Test-chip Implementation

A test-chip in 0.13m CMOS demonstrates the single-chip image sensor (Figure 3.16(a)). The chip is wire bonded in open cavity LCC 64 package. The inductor of the power management unit is off-chip, and integrated on the PCB. Key parameters of the test-chip and

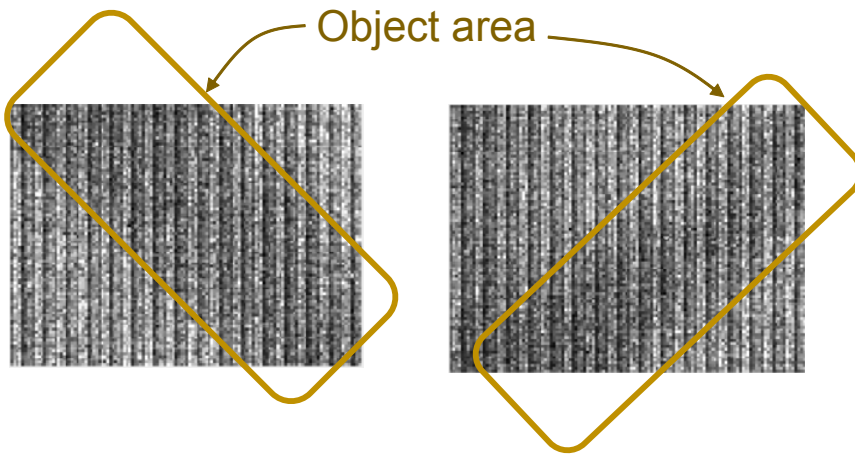


Figure 3.17: Images captured with a thin object in front of the sensor.

power management unit are listed in Figure 3.16(b) and (c).

Functionality of the Image Processing Engine

In this subsection, I evaluate the functionality of the image processor. Figure 3.17 shows images captured with a static thin object in front of the sensor in different positions. The image shows high level of random and fixed-pattern noise. I will discuss later the cause of the noise and how I plan to reduce the noise. Also, the pixel value differences between the object (dark) and the background (bright) are not significant, indicating very limited dynamic range. The image processor operates on the noisy image and selects the ROI MBs to transmit.

I evaluate the functionality of the image processor with the noisy image from the sensor array. First I apply a static scene to the sensor and count the number of transmitted MBs by varying the threshold value [Figure 3.18(a)]. Ideally, with static scenes no blocks should be transmitted when the threshold is larger than zero. However, the input image is dominated by the noise generated at the sensor, which results in edge movement (high activity level) at the entire region of the image. Therefore, a large number of blocks are transmitted even without actual motion of objects. The number of transmitted blocks matches with the simulation, indicating that the image processor functions as desired. I also perform

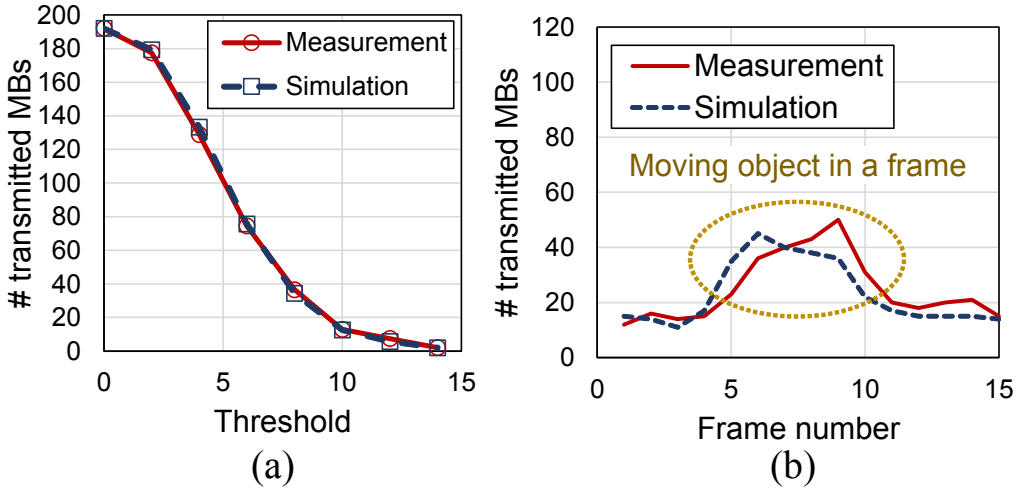


Figure 3.18: (a) The number of transmitted MBs with varying threshold. (b) The number of transmitted MBs over frames with a moving object.

the experiment with motion by moving a thin object in front of the sensor. Figure 3.18(b) shows that more MBs are transmitted in frames with a moving object, and the number of transmitted blocks fairly matches with the simulation result. With the threshold value of 6, average 23 MBs are transmitted for 21 frames, which achieves 8.2X transmission energy reduction.

System Self-Power Performance

A cool white (7000K) LED lamp 1000 lm is used for photo response measurement. It generates 2.1W of peak power at 200klx luminance. Based on measured power generation and measured power consumption of the sensor, I estimate the self-supported frame rate (frame/sec). Figure 3.19 shows the consumed/harvested energy over time. At the maximum operating frequency of the system, processing of one frame (image sensing, processing, and preparing packets for transmission) consumes 13.9J of dynamic energy. After transmitting the last block of the frame, the system switches into the harvesting mode. During the harvesting mode, the entire system components except for the image sensor array, the voltage regulator, and the SRAM are clock-gated and power-gated to avoid the leakage

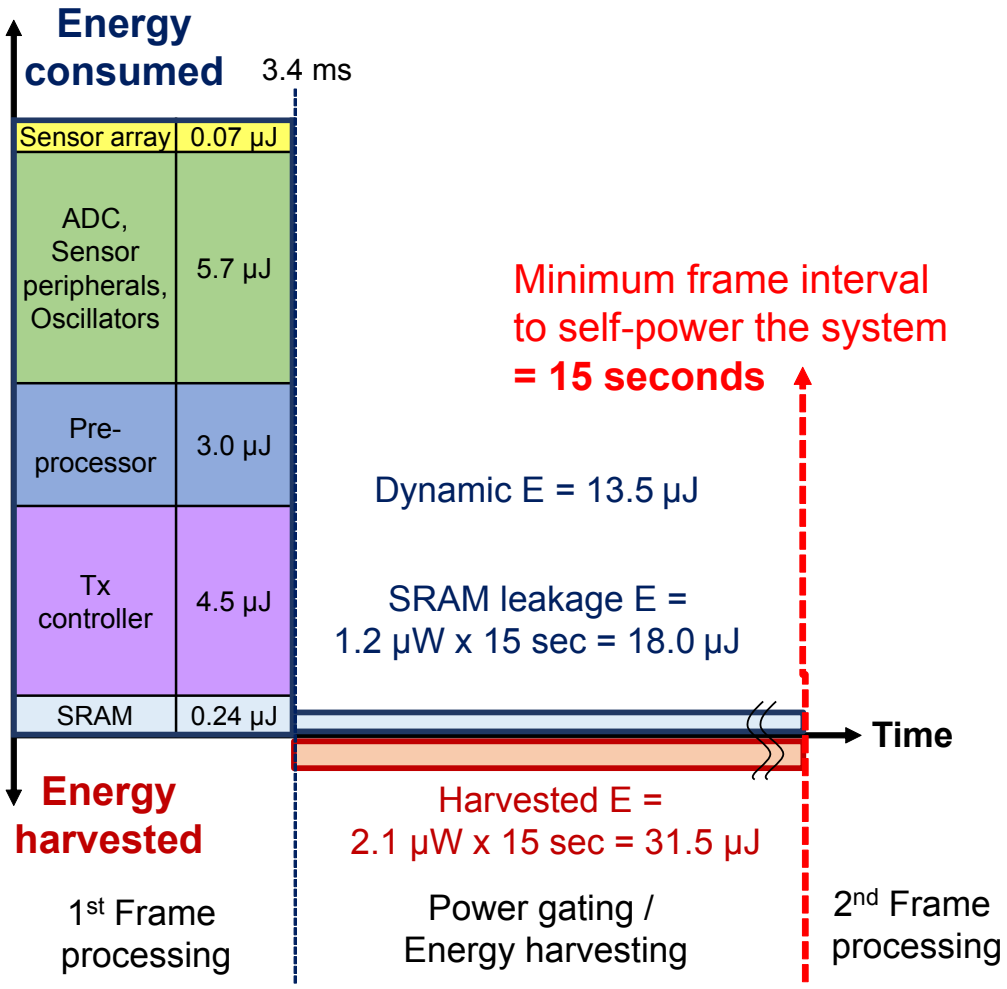


Figure 3.19: Diagram showing the consumed/harvested energy over time.

energy consumption. The SRAM should remain powered-on since the edge map of the previous frame stored in the SRAM needs to be loaded during next frame processing for moving object detection. When supplied with the nominal voltage of 1.2V, the SRAM consumes leakage power of 1.2W. The image sensor array in the harvesting mode generates 2.1W assuming 200 klx light intensity. Therefore, during the frame idle period, the net power gain that can be stored in the battery is the difference between the harvested power and the SRAM leakage power, which is 2.1 - 1.2 = 0.9W. This power gain should be integrated over time to supply the dynamic energy of the system for frame processing (13.5J). In this setup, the minimum frame interval for self-power operation is 15 seconds, which

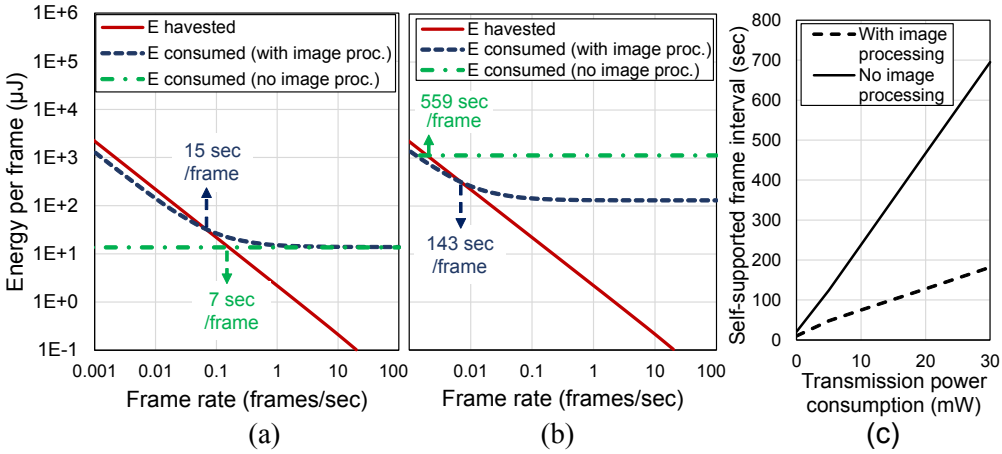


Figure 3.20: Harvested/consumed energy for varying frame rate (a) without the wireless transmission and (b) with the wireless transmission by nRF transmitter, (c) self-power performance vs. transmission power consumption.

leads to 4 frames/minute.

To project the self-power performance under varying frame rate, I consider two configurations of the system. Figure 3.20(a) shows the frame rate when the harvested energy powers only the on-chip system (assuming the wireless transmitter is powered by a separate energy source). Figure 3.20(b) is when off-chip wireless transmission is also powered by energy harvesting, which will be discussed in Section IV.B. Figure 3.20(a) shows that, as the frame rate decreases, harvested energy increases because of the increased idle time. However, when image processing is enabled, energy consumption also increases with lower frame rate since the leakage energy of the SRAM increases with the frame idle time. One may decide to disable the image processing to avoid SRAM leakage energy during the harvesting mode. If the image processor and the SRAM are turned off (transmitting all the MBs), the minimum frame interval decreases to 7 seconds. However, turning off the processing will increase the data volume, and hence, as I show later, will reduce the self-powered frame rate when a wireless transmitter is integrated with the system.

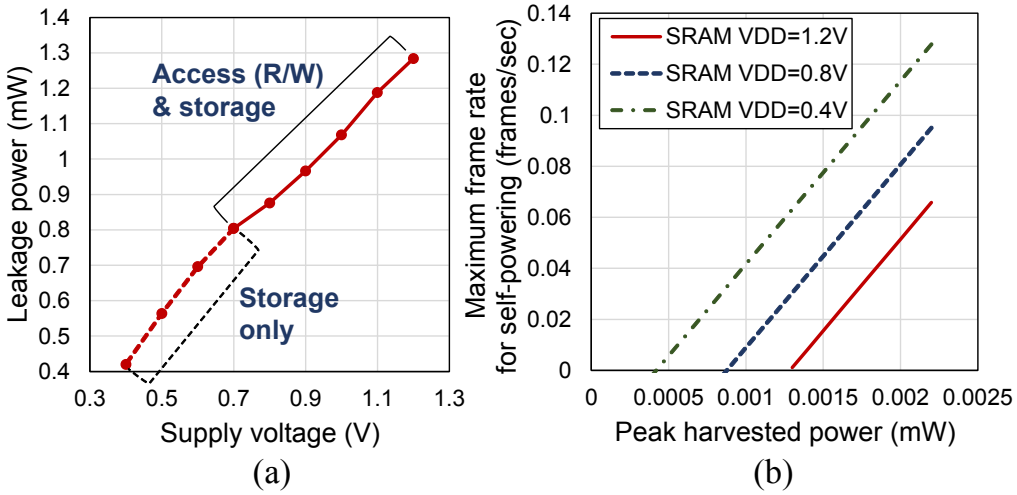


Figure 3.21: (a) SRAM leakage power vs. supply voltage. (b) Maximum frame rate for self-powered operation with different SRAM supply voltage.

3.2.4 Approaches to Improve Self-Power Performance

Effect of SRAM Leakage Energy

As discussed in the previous section, SRAM leakage power is a key component that determines the system self-powering performance. One way to reduce the SRAM leakage power is to operate it at lower supply voltage. To understand the minimum voltage at which SRAM can function and/or retain data, I perform the experiment by varying the supply voltage. As Figure 3.21(a) shows, for our design, the supply voltage below 0.7V does not guarantee the correct functionality of reading and writing. However, the SRAM can retain data down to 0.4V of operating voltage. Therefore, we can use 1.2V supply for SRAM (same as digital logic) during frame capture/processing to enhance processing speed and avoid level shifters. In the idle mode, we can reduce the supply down to 0.4V. To enable the adaptive supply control, I propose to separate the SRAM power supply from the logic engine. Leakage power reduction through adaptive supply voltage control enhances the self-power performance. Figure 3.21(b) shows that supplying 0.4V during the harvesting mode increases the self-supported frame rate (no wireless transmission) to 0.13 frames/sec,

which is 2X higher than the frame rate at 1.2V.

Effect of Wireless Transmission

In a wireless sensor node, transmission energy is usually the major component of the system energy. I simulate the self-power performance considering the presence of the nRF24L01+ transmitter in the system. This commercial radio chip consumes 37.3mW at 0dBm and 23.1mW at -18dBm output power. In this simulation, I assume -18dBm output power. Also, I assume the reduction of transmitted data by the image processor is 8.2X, which is obtained through the experiment with a moving wire in Figure 3.17-3.18. As expected, if we consider the wireless transmitter is powered by the harvested energy [Figure 3.20(b)], the minimum frame interval increases significantly due to transmission energy of the transmitter. Disabling image processing even increases the self-supported frame interval due to increase in the transmission energy, which overshadows the SRAM leakage elimination. I observe that the estimated self-power performance considering the transmission energy is 143 seconds/frame and 75 seconds/frame with 1.2V and 0.4V power supply for SRAM, respectively. When the system with wireless transmitter is self-powered, the self-power performance is largely dependent on the power consumption of the transmitter. Recent advancements in radios for IoT sensor nodes have focused on enhancing transmission power efficiency. Reduced transmission power enhances self-power performance, as illustrated in Figure 3.20(c). The technology trend [78] shows that the current state-of-the-art on-chip radio consumes about 5mW at 0dBm output power, which is about 5 times lower than the off-chip radio used in the current simulation. By using 5mW power consumption for the projection of self-power performance, the minimum frame interval reduces to 124 seconds (without image processing) and 56 seconds (with image processing). The system with image processing can transmit one frame every minute, which makes it more applicable to mobile surveillance applications.

Effect of Unit Pixel Size

As presented in the previous sections, the self-powered frame rate is limited by harvested power from the sensor array. One of the reasons of low harvested power is extra metal layers at the pixel boundaries. These metal layers are placed because of the minimum metal density restrictions of the foundry, causing a decrease in the amount of pixel area exposed to light. Therefore, increasing unit pixel area can increase the light exposure, thus, enhance the energy harvesting performance. Increased pixel area is also expected to enhance the dynamic range of the image sensing. The major disadvantage of increasing unit pixel size is the reduced image resolution per array area. If we want to keep the same number of pixels (resolution), we need more area for the sensor array. Similarly, to keep the array area same, we need to reduce the number of pixels in the array. Lower image resolution results in lower perceptual quality to the users. However, lower resolution does not significantly degrade the detection performance of the moving object detection method. Lower resolution has an advantage in system energy consumption because it reduces the computation energy and transmission energy per frame. Also, it requires smaller memory for the edge map, reducing memory leakage energy. With 4X increase in unit pixel area, the number of pixels decrease by 4X, so we can roughly estimate the system energy to be 1/4X. This resizing increases the fill factor of the pixel from 44% to 69%. Assuming the harvested power linearly increases with the fill factor, we can expect the peak harvested power to increase from 2.1 to 3.3W. With the increased harvested energy and reduced system energy, the self-power performance enhances from 559 seconds/frame to 137 seconds/frame with image processing, and from 143 seconds/frame to 25 seconds/frame without image processing.

Effect of Power Converter Efficiency

Self-power simulation results presented above assume 100% efficiency of the power converter. Therefore, the self-power performance is expected to be degraded with lower efficiency value. At an input source power of 2.76W the boost converter shows 24% efficiency

(simulated considering all parasitics). I found that this low efficiency is mainly due to switching losses in the power stage MOSFETs. Since the power converter uses only one power stage, it cannot be optimized simultaneously for two widely varying power ranges (W range during boost, mW range during buck), thus leading to the degraded efficiency for low input power. A possible solution for this would be to split up the power converter into boost and buck stages this would allow us to improve efficiency by decoupling the buck and boost stages, and thus optimally size them to obtain higher efficiency in their typical operating ranges. Using a boost converter with power stage sized for W power range provides us with a simulated efficiency of -80%.

Summary

The preceding discussions shows the reducing SRAM supply voltage, using low-power on-chip radio, increasing pixel size, and re-designing the PMU to independently optimize the boost and buck stage provides opportunities to enhance self-power performance. The estimated self-power performance of the test-chip based on measurements reported in section III and assuming nRF transceiver was 143 seconds/frame. Considering 80% efficiency of the boost converter the self-power performance drops to 271 seconds/frame. Now, if we apply 0.4V of SRAM operating voltage the self-power performance improves to 102 seconds/frame. Further, if we consider the transceiver to be replaced by a low-power on-chip radio the self-power performance improves to 34 seconds/frame. Finally, the pixel sizes are increased for same area, the performance improves to 20 seconds/frame. The analysis shows that the proposed system can be designed to approach a reasonable self-power performance for surveillance applications.

3.2.5 Summary of the Section

This section presented a single-chip image sensor node that can harvest energy from the on-chip pixel-array. The reconfigurable dual-purpose image sensor performs as an imager

as well as an energy harvesting transducer, producing the peak harvested power of 2.1W. The moving object detection method with low computation and memory demand enhances the self-power performance by reducing unnecessary transmission of the static background. Low-power circuit techniques such as block-level pipelining, power gating, and adaptive supply control are utilized to further improve the self-power performance. The measurement results, although shows noisy image, demonstrates the feasibility of a self-powered sensor with operating performance of 7 seconds/frame and 75 seconds/frame with and without wireless transmission, respectively. In conclusion, the proposed system is an important step towards energy autonomous imaging applications for various IoT applications such as remote surveillance and capsule endoscopy.

CHAPTER 4

RESOURCE-EFFICIENT AND ROBUST IMAGE PROCESSING

4.1 Low-Power Noise-Robust Moving Object Detection

4.1.1 Introduction

The previous chapter presented hardware implementation of an image sensor node. One of the challenges in the the image sensing from the sensor node is random noise induced from the image sensor array, which can corrupt the captured images. In addition to the random noise, in their outdoor applications, the sensor platforms are often exposed to dynamic environment, where objects of interest usually move amidst noisy backgrounds, for example, snow, rain, etc. [79]. Even under such dynamic conditions, moving object detection methods are anticipated to perform reliable ROI detection in order to reduce unnecessary waste of channel bandwidth and energy for transmitting background scenes. Several studies have proposed moving object detection methods that are robust to dynamic (noisy) environment [12][13][14][15][17]. However, as I discuss later, the existing methods require large energy and area for memory and computation, and are generally not scalable to resource-constrained systems. On the other hand, the moving object detection methods developed for low energy/area overheads are often not robust under dynamic environment. Therefore, there is a need for a moving object detection method that is suitable for compact and low-power systems while tolerant to noise induced by the dynamic environment [8][9][10][11]. This section presents an energy-aware approach to noise-robust moving object detection for resource-constrained sensor platforms. The key contribution is the design of a block-level rank closing operation to improve noise robustness of the existing moving object detection method using sequential edge detection and frame differencing (ED+FD). The proposed method is integrated with a block-based processing unit and the

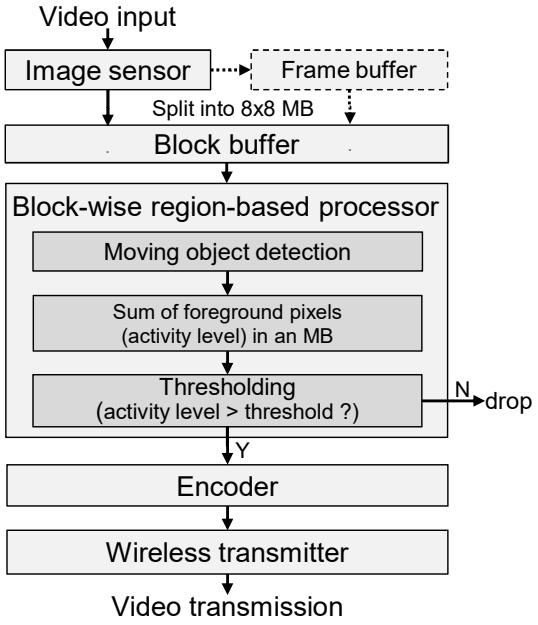


Figure 4.1: A wireless image sensor platform with a block-wise region-based processing model.

motion JPEG encoder to implement the image-processing pipeline of a wireless image sensor platform. The sensor platform is designed as an ASIC in 130nm CMOS for energy/area analysis, as well as prototyped into Virtex-5 FPGA for functional validation. The proposed approach provides comparable performance with the existing noise-robust method based on Gaussian Mixture Model (GMM), but consumes 28X and 2.7X lower area and processing energy, respectively. The ASIC (130nm CMOS) realization shows only 2.1% energy overhead compared to the whole system; however, the system-level analysis shows significantly lower energy at the same quality of ROI. The primary advantage comes from the significant reduction in the on-chip memory capacity/energy. If off-chip memory is considered, our approach significantly reduces off-chip memory access rates, and hence, memory bandwidth requirements and access energy. In summary, as the proposed system has very low computation and memory requirements, it can be applied to a resource-constrained

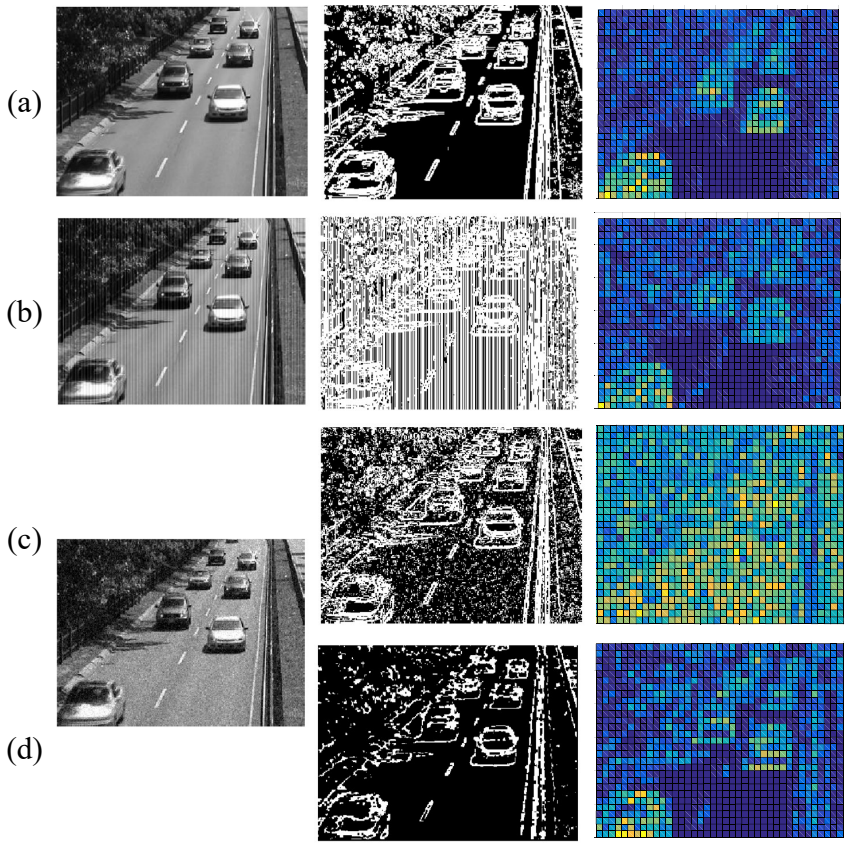


Figure 4.2: (left): input images, (middle): edge map, (right): block-level activity map with (a) original video, (b) with FPN, (c) with random noise using the original ED+FD method. (d) images with random noise using the proposed noise-robust method.

wireless image sensor platform.

4.1.2 Effect of Noise on the ROI Processing

Sensor-Induced Noise

In this subsection, I discuss algorithmic approaches to enhance the performance of moving object detection under pixel-noise. I consider both fixed-pattern noise and random noise to perform the analysis. I add random and/or fixed pattern noise to a benchmark video sequence (highway) [Figure 4.2(a)] to study the performance of in moving object detection using the ED+FD method under noise. I also explore other complex approaches with

enhanced noise robustness based on Optical Flow (OF) [17] and Gaussian Mixture Model (GMM) [12]. As for the ED+FD (Edge detection + frame differencing) and the proposed method (ED+FD+noise robustness), the algorithms are designed as RTL and synthesized into the IBM 130nm process. The synthesized designs are supplied to the Synopsys Prime-time tool to determine power consumption and latency. To obtain the computation energy of OF and GMM methods, I consider the reference designs in prior works [17][12]. In these studies, the designs have been synthesized into the 90nm process. Therefore, I scaled the computation energy described in the papers into the 130nm process using the scaling parameters from International Technology Roadmap for Semiconductors [80]. Also, the memory access and storage requirements mentioned in the paper are translated into the memory energy by multiplying the unit memory access and leakage energy of SRAM implemented in the 130nm process.

Fixed-Pattern Noise FPN can be modeled by adding a single value per column of pixels in the array. The added column values follow the Gaussian distribution and remain fixed over multiple frames [81]. Figure 4.2(b) shows an image with FPN of the variation=0.05. Although FPN creates vertical lines in the original image and edge map, their locations do not change over frames. Therefore, frame differencing of two consecutive edge maps removes most of the lines generated by FPN. As a result, increase in the activity levels at the background due to FPN is not significant, generating similar distribution of the activity map as the original image. Therefore, it can be claimed that certain level of FPN can be effectively handled by the nature of ED+FD method. However, as the variance increases, the original image degrades more. Hence, false detection increases, resulting in lower data reduction for a given ROI delivery ratio. Here I compare the detection performance with existing complex moving object detection methods based on OF and GMM. ED+FD shows comparable noise-robustness to the complex methods such as OF and GMM.

Random Noise Random noise can be modeled as a zero-mean Gaussian distribution [82]. If the pixels in an input image are affected by the random noise, their boundary pixels will be determined as edges in an edge map. As the location of noisy pixels changes frame by frame, the edge locations also change, resulting in large activity level at the background due to noise [Figure 4.2(c)]. As the random noise affects individual pixels, it can be removed by pixel-level low-pass filtering such as median of average filtering. However, these filtering techniques can result in significant computation overhead. For instance, filtering of n x n image with 3x3 filter requires 9-element sorting operations for n2 points. This computation demand will translate into energy and latency overhead. In addition, although low-pass filtering removes impulse noise, it leads to blurring of the original image regardless of the existence of noise, resulting in quality degradation. The goal in this section is to reliable detection moving object under noisy environment, not the noise elimination in the original image. Therefore, I propose an approach that adds a simple operation to the ED+FD process, instead of altering the original image. In a baseline ED+FD method, the ROI is determined by the activity levels of the blocks, which is computed by inter-frame difference of edge maps. Therefore, I propose removing the edges generated by impulse noise in each edge map, instead of filtering the original image. To remove impulse edge pixels in the edge map, I use a simple median filter with 3x3 kernel size. Median filtering on an image generally requires sorting elements in a kernel, which needs computation of O(nlogn). However, as we perform filtering on a bit-map of edges, it can be implemented simply by adding the bits in the kernel and comparing the value with the threshold (5 for median filter).

Figure 4.2(d) shows the edge map after the proposed filtering process. It shows that most of the small group of edges generated by random noise are removed, resulting in the similar activity map as the original video. Figure 4.3(a) shows the performance of the original ED+FD method significantly degrades since random noise increases activity level (movement of edges) at the background. However, after performing edge map filtering, the

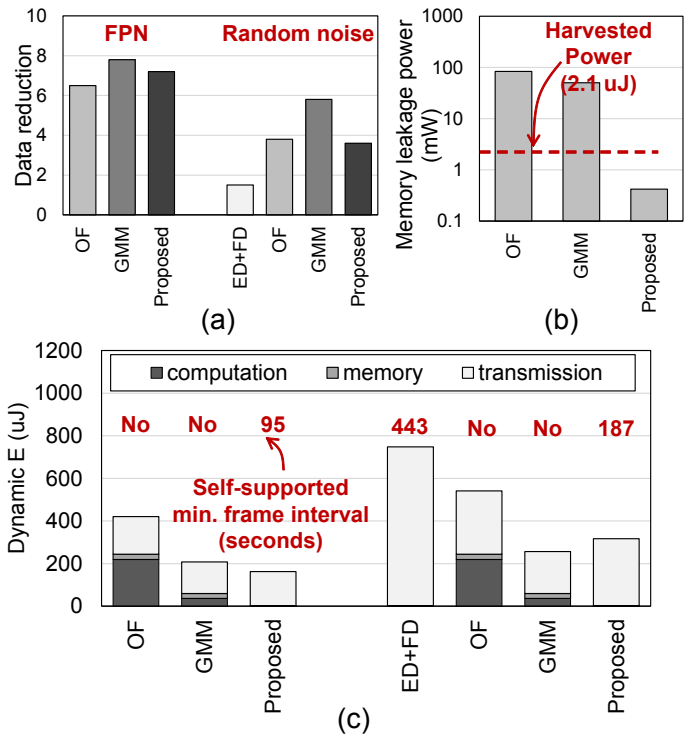


Figure 4.3: (a) Data reduction through different moving object detection methods under FPN and random noise. (b) Memory leakage power of the methods. (c) Dynamic energy and self-supported minimum frame interval of the methods under FPN and random noise.

non-ROI activity level significantly decreases, thereby reducing false detection. Although OF and GMM outperforms the proposed method, they require significant memory size (OF: 200 bit/pixel, GMM: 120 bit/pixel). As a result, the leakage power of the SRAM for those methods is much higher than the harvested power at the maximum brightness [Figure 4.3(b)]. Therefore, the system with those high memory demand moving object methods cannot be self-powered by energy harvesting. A low memory requirement (storage: 1 bit/pixel, access: 2 bit/pixel) of the proposed moving object detection method enables the self-supported operation with a wireless transmitter at frame rate of 95 seconds/frame when FPN exists [Figure 4.3(c)]. In case of random noise, the self-power performance of the original ED+FD method significantly degrades because of the large transmission energy consumption due to the false detection. By reducing the false detection through noise robustness, the proposed approach can be self-powered at much higher frame rate than

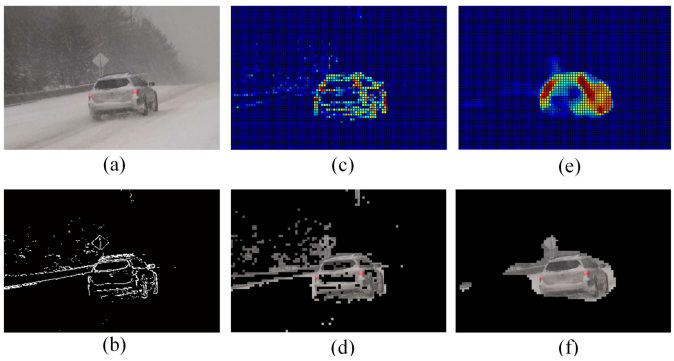


Figure 4.4: (a) Original snowfall video frame. (b) Edge map generated by ED. (c) Block-level activity map and (d) transmitted image using ED+FD. (e) Block-level activity map and (f) transmitted frame image using the proposed method.

ED+FD.

Environmental Noise

The goal of this subsection is to enhance the robustness of ROI detection against the dynamic weather condition with minimum area/energy overheads. As an underlying moving object detection method, I consider the ED+FD approach presented in section . In this approach, the edge maps of each MB for two subsequent frames are compared, and the MBs with the sum of the pixel-wise difference (i.e., activity level) larger than the threshold are detected as ROI. Here I use a benchmark video sequence with a bad weather environment of a snowing scene (snowfall) to explain the challenges in ROI detection using the ED+FD method under noise. Figure 4.4(a) shows a frame in the snowfall video. The major difficulty in detecting moving objects from snowing scenes is the movement of snowflakes. For better performance, one should not detect snow particles as moving objects. However, the original ED+FD method is not effective in dealing with snowing scenes because snowflakes are detected as moving edges [Figure 4.4(b)]. Therefore, the activity levels of MBs with snowflakes increase, as illustrated in Figure 4.4(c). Consequently, as Figure 4.4(d) shows, the original ED+FD falsely detects the background locations with snowflakes as moving objects. If background MBs are falsely detected as the ROI and encoded/transmitted, some

of the real ROI MBs can be randomly dropped to meet the wireless channel bandwidth, resulting in lower quality-of-service. This effect is illustrated quantitatively in section 3.4.

4.1.3 Proposed Approach

Basic Concept of the Proposed Approach

The preceding discussion suggests that the noise-robust detection method should be able to distinguish between moving objects of interest and other sources of noise; for example, the snowflakes in Figure 4.4. Usually, a major difference between them is the size; objects of interest usually extend over multiple MBs while snowflakes cover only one or two MBs. Therefore, I hypothesize that, if the high activity-level is observed in a MB surrounded by low-activity MBs, the higher activity of the MB is due to noise. Therefore, we can reduce the false detection by applying spatial de-noising filtering to the block-level activity map. This is likely to smooth out the high ED+FD scores that are surrounded by low ED+FD scores. Since I propose to apply a spatial de-noising filter at the block level, not on the pixels, the cost of filtering is much lower compared to pixel-level filtering.

Noise-Robust ROI Detection Method

The proposed method is illustrated in Figure 4.5. The first part of the flow, which is identical to the original ED+FD method, generates the block-level activity map (ED+FD score map of MBs). I propose to apply a rank closing operation, which is two rank-order filtering operations (one with a high rank and one with a low rank), on the block-level activity map to generate the filtered activity map. The filtered activity value of each MB is compared with the threshold to make a decision on the MB.

The output value of a rank filter of a rank k with a window B of size $n(\zeta_{B,k})$ is obtained by sorting the pixel values within the window in ascending order and selecting the kth value in the sorted array. In the block-level activity map, only a small proportion of MBs within a window are likely to be noisy MBs, and they tend to occupy the extreme rank positions.

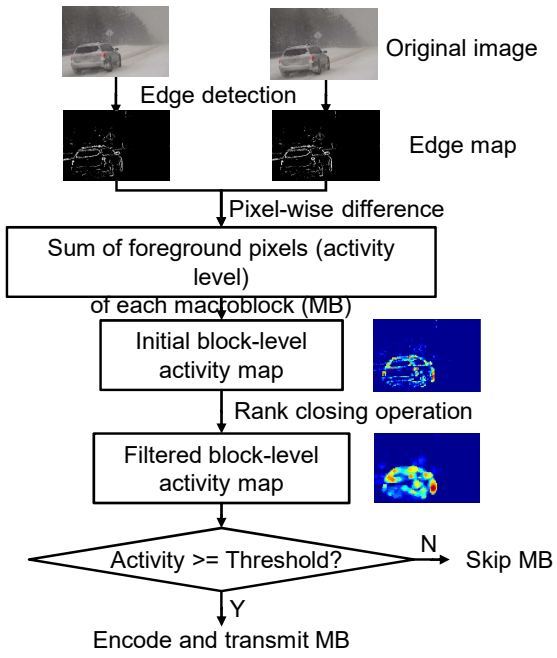


Figure 4.5: A flowchart of the proposed detection algorithm.

Therefore, if a rank filter with a rank k such that $n/2 < k < n$ is used, these impulse activity level values will not be selected. While the rank filtering operation eliminates impulse noise, it is subject to expand edges of objects, as shown in Figure 4.6(b). Therefore, it should be followed by another rank filtering operation with a low rank to maintain the original structure by shrinking the boundary. These two rank filtering operations benefit the detection performance especially when they are combined with the ED+FD method. As ED+FD is based on the edge information, its major drawback is that it is not effective in detecting the regions inside of moving objects when their sizes are large or inside regions are not sufficiently textured. Therefore, for better performance of the ED+FD method, it should be followed by a morphological operation that can fill inside of such a region. One of such operations is morphological closing, which closes gaps between structures without expanding the size of the structures. The closing operation (φ_B) is defined by a dilation (δ_B) followed by an erosion (ε_B);

$$\varphi_B = \varepsilon_B \delta_B \ . \tag{4.1}$$

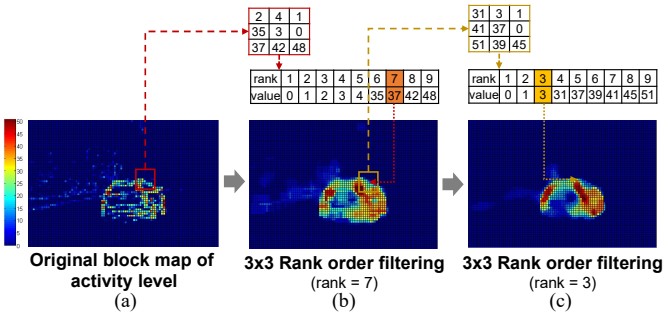


Figure 4.6: Example of rank closing operation. (a) Original block-level activity map. (b) First rank filtering with a high rank. (c) Second rank filtering with a low rank.

The erosion operator is equivalent to a rank filter of rank 1, and the dilation to a rank filter of rank n, where n is the number of elements in the window B;

$$\varepsilon_B = \zeta_{B,1}, and \delta_B = \zeta_{B,n}. \tag{4.2}$$

We obtain the rank closing ($\varphi_{B,k}$) by replacing the erosion and dilation with general rank operators as shown below [16];

$$\varphi_{B,k} = \zeta_{B,n-k}\zeta_{B,k} \; ; \; \frac{n}{2} < k < n. \tag{4.3}$$

The dilation with a rank k¡n removes impulse noise. Also, the dilation grows the object and fills gaps, and the erosion shrinks the structure. Therefore, the regions expanded towards outside the boundary by the dilation will shrink by the erosion, while objects merged by the dilation will not be separated again by the erosion. Therefore, if rank closing is applied on an image, small holes inside objects are filled, and impulse noise is removed while keeping their structure. Although rank closing is a well-known algorithm for morphologic operation, our approach is novel in the sense that it is applied to the block-level activity map instead of the original image in order to enhance the noise-robustness of moving object detection. When it is applied on the block-level activity map, the regions with high activ-

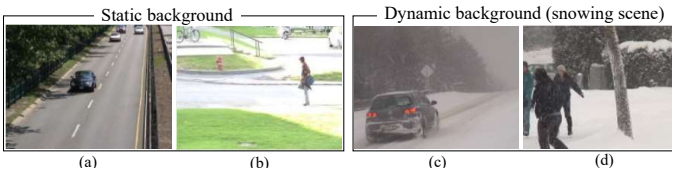


Figure 4.7: Original frame images of test video sequences. (a) highway, (b) pedestrian, (c) snowfall, and (d) skating.

ity levels inside the object expand to merge together, while sparse MBs with high activity levels are removed [Figure 4.6(c)]. While many studies have proposed similar morphological operations [83], most approaches are computationally complex, and they require large memory. Conversely, rank closing is relatively low-overhead because it is a combination of two simple rank order filtering operations. Also, the added overhead to the original ED+FD method is negligible since rank closing is performed on the block-level activity map, which is 64x smaller than the original frame size. For storing the block-level activity map, it requires only 6 bits/MB, i.e. 6 bits/64 pixels (0.09bit/pixel compared to 120-200 bits/pixel in GMM) [Figure 2.1(a)].

4.1.4 Analysis of Detection Performance

For a comparative performance evaluation of moving object detection methods, I develop a MATLAB/Simulink-based simulation framework that includes the proposed method, ED+FD, OF, and GMM. Our GMM model uses three Gaussian distributions, and OF is based on the Horn-Schunck method. I use four video sequences from the 2014 dataset in changedetection.net [84]; highway and pedestrian from the baseline category, and snowfall and skating from the bad weather category [Figure 4.7]. The ground-truth ROI MBs (i.e. real MBs with moving object in each frame) are determined based on the pixel-wise ground-truth ROI information provided by the database.

As explained before, the original ED+FD method makes false detection on the background because snowflakes are detected as moving edges, which result in higher activity

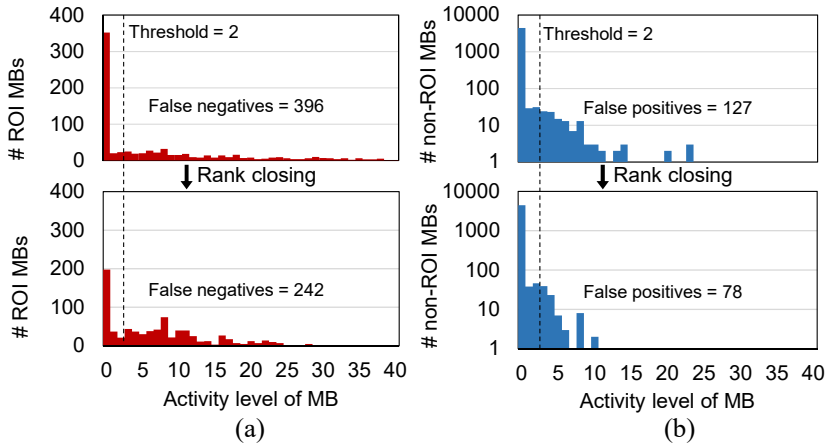


Figure 4.8: Activity level distribution of the ground-truth (a) ROI MBs and (b) non-ROI MBs before and after rank closing (with snowfall video).

levels in such a region [Figure 4.4(c, d)]. Figure 4.4(e) shows the block-level activity map filtered by the rank closing operation; the regions with high activity levels are expanded to fill the inside of the object-of-interest, while the activity levels of noisy MBs decrease. The effect of the rank closing operation is quantitatively illustrated in Figure 4.8. With rank closing, the ROI MBs with zero-activity decreases, and the activity-level distribution of non-ROI moves towards zero, resulting in higher delivery of ROI with more reduction of non-ROI for the same threshold value. Hence, with the rank closing operation added to the ED+FD method, noisy MBs are discarded, and more MBs inside the moving objects are transmitted [Figure 4.4(f)].

Figure 4.9(a) shows the performance of each method indicating how many ROI MBs are delivered for given data reduction according to the varying threshold values with snowfall video. Figure 4.9(b) presents data reduction achieved by methods when ROI delivery is 0.8 for each video sequence. The figure indicates that ED+FD yields comparable performance to sophisticated ones such as GMM when dealing with static background videos. On the other hand, under bad weather condition, data reduction of ED+FD becomes much

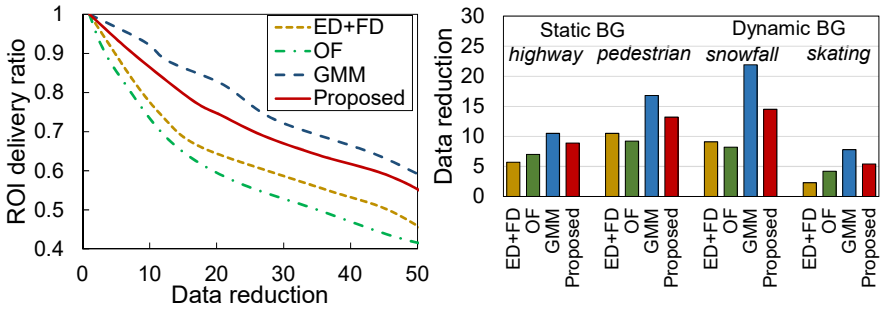
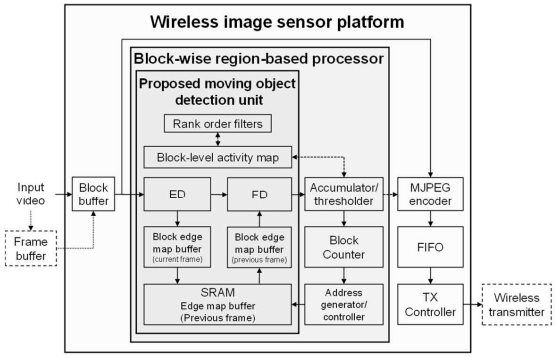


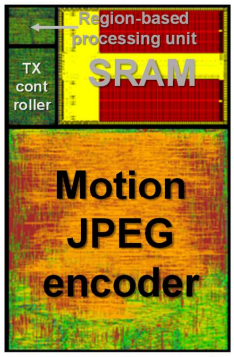
Figure 4.9: (a) Data reduction vs. ROI delivery ratio (snowfall) and (b) data reduction @ROI delivery=0.8.

lower than GMM. However, with enhanced noise-robustness, the proposed approach significantly improves detection performance (1.6X and 2.3X more reduction and ED+FD for snowfall and skating). Note the proposed approach improves the performance even with static background because more adjoining MBs inside the objects-of-interest are detected together as groups due to the rank closing operation. In summary, the proposed approach (ED+FD+rank closing) shows comparable performance as GMM; but with a much lower memory and computation requirement.

4.1.5 Application to a Wireless Image Sensor Platform

The block-wise image processing engine with the proposed noise-robust moving object detection method is integrated with a wireless image sensor platform. For analysis, I consider an illustrative wireless transmitter by Nordic Semiconductor (nRF24L01+) [73]. I assume input video size of 320x240. The block diagram of the processing engine is shown in Figure 4.10(a). The MBs can be fed directly from an image sensor or from a frame buffer. Each MB in a frame is processed serially by the proposed moving object detection method with ED, FD, and rank closing. Based on the decision signal generated by the detection unit, the block is either dropped, or encoded by the MJPEG encoder and transmitted by the off-chip wireless transmitter. I use the MJPEG encoder in this system since the MJPEG





Area and Energy Analysis in 130nm CMOS				
Module			Area (mm^2)	Energy (uJ/frame)
Region-based processor	Moving Object Detection unit	Computation unit	0.021	0.55
		Memory (SRAM)	0.49	1.65
		Total	0.51 (25%)	2.2 (2.1%)
	Thresholder, counter		0.002	0.032
	Total		0.512	2.23
MJPEG (variable QF†)			1.41	78.4
Tx controller			0.064	23.9
Total			**1.98**	**104.5**

†Fixed-QF MJPEG: 1.02 mm^2, 68.7 uJ/frame

(c)

Figure 4.10: The hardware implementation: (a) block diagram of the wireless image sensor platform, (b) the ASIC implementation (layout) of the platform, and (c) area and energy analysis (130nm CMOS).

requires much less resources than complex encoders like H.264 and MPEG, and hence, more suitable for resource-constrained platforms. The standard MJPEG yields lower compression ratio as it does not exploit temporal dependency. However, the MJPEG with a moving object detection scheme that exploits temporal dependency between frames can provide better quality than H.264/intra under the same energy and area [11]. The system is implemented in ASIC with 130nm CMOS [Figure 4.10(b), 4.10(c)]. The ASIC implementation operates at 128 MHz (1.5V), supports 40 frames/sec, and consumes processing energy of 104.5 J/frame. The proposed noise-robust moving object detection method occupies only 25% of area and 2.1% of energy of the entire processing engine [Figure 4.10(c)]. The overhead to the overall system energy is much lower since the system energy includes transmission energy as well.

System-level Energy and Performance

Table 4.1 describes the energy consumption model for each component of the sensor platform. The energy associated with processing and memory is obtained from the simulation results of the ASIC implementation. The computation energy and area of OF and GMM methods are scaled to the 130nm from prior works [17][85] using the scaling parameters

Component		Energy (nJ)	Note
Processing	Detection unit (computation)	$p \cdot e$	*p=number of pixels per frame* *e=computation energy/pixel* *(ED+FD:0.007, OF:0.51, GMM:0.086, Proposed:0.0072)*
	Detection unit (memory)	$\frac{0.137 \cdot p \cdot s}{f}$	*s=required bit storage per pixel* *(ED+FD:1, OF:195.5, GMM:120, Proposed:1.09)* *f=frame rate*
	MJPEG	$\frac{1.02 \cdot p}{r}$	*r=data reduction*
	Tx controller	$\frac{0.311 \cdot p}{r \cdot c}$	*c=compression ratio*
Transmission		$11.2 \times \frac{p \cdot 8}{r \cdot c} + 900\left(\frac{1}{f} - T_p\right) + E_s$	T_p*=frame processing time* E_S*=settling energy* *(0 if frame data size·*T_P*≥BW, 5578 otherwise)* *Condition: Tx bandwidth=2Mbps, signal power=-18dB, payload=32B*

Table 4.1: Energy consumption model

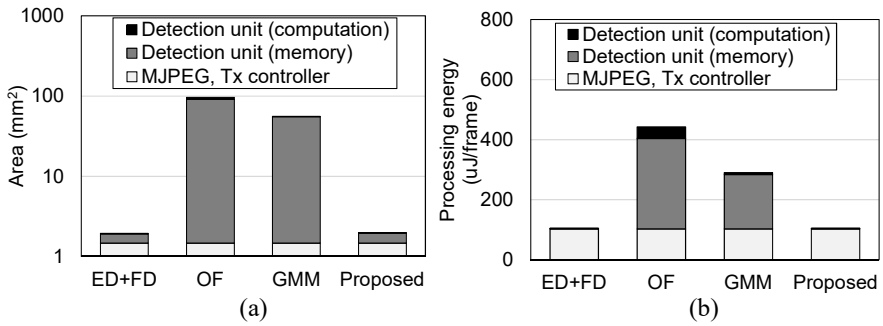


Figure 4.11: (a) Area and (b) energy consumption of the platform equipped with each moving object detection unit.

from International Technology Roadmap for Semiconductors [80]. Transmission energy is modeled from the data sheet of the nRF transmitter.

Figure 4.11 shows the area and energy of the systems with four different moving object detection methods. While the proposed approach incurs small area and energy overhead to the system, OF and GMM occupy significant portion of the total system area and energy, mainly due to their large memory requirements. For example, GMM consumes 97% and 65% of the total system area and energy, respectively.

For evaluating the energy-efficiency of the systems with different detection methods, I plot ROI quality versus system energy (computation + transmission energy) [Figure

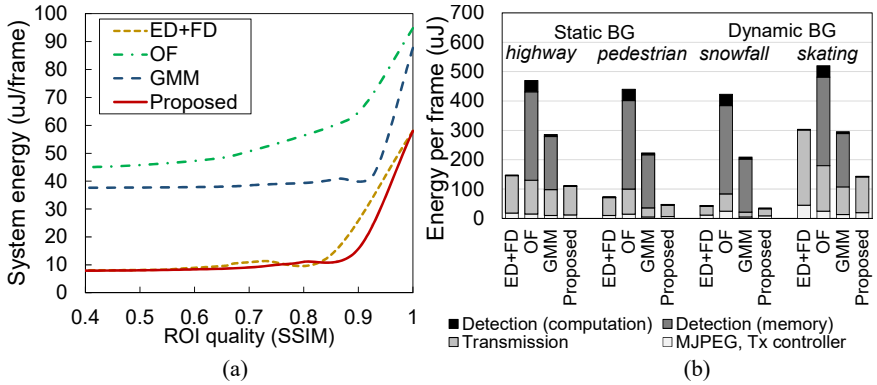


Figure 4.12: (a) System energy vs. ROI quality (snowfall) and (b) energy consumption breakdown @ROI quality=0.8.

4.12(a)]. ROI quality is expressed with the structural similarity (SSIM) index of the ground-truth ROI MBs when the quality factor of the MJPEG is fixed at 50. I control ROI quality by changing ROI delivery ratio, which is determined by the threshold value in the region-based processing unit. Figure 4.12(b) shows the energy component breakdown of the system at the ROI quality (SSIM) of 0.8. Although GMM is shown to be the most effective method in reducing the amount of data for a given delivery ratio [Figure 4.9], it does not show a good system-level energy-efficiency because of its large memory requirement and hence, storage energy. The original ED+FD provides better efficiency than GMM for the video sequences with the static background. However, its energy-quality characteristic degrades for the bad weather sequence (skating) since it consumes more transmission energy to unnecessarily deliver the background data. By reducing the transmission energy through the low-overhead noise-robust technique, the proposed method outperforms other methods in all four sequences. Although the proposed approach consumes more energy for transmitting data than GMM, its total system energy is less than GMM due to its low computation and memory requirements.

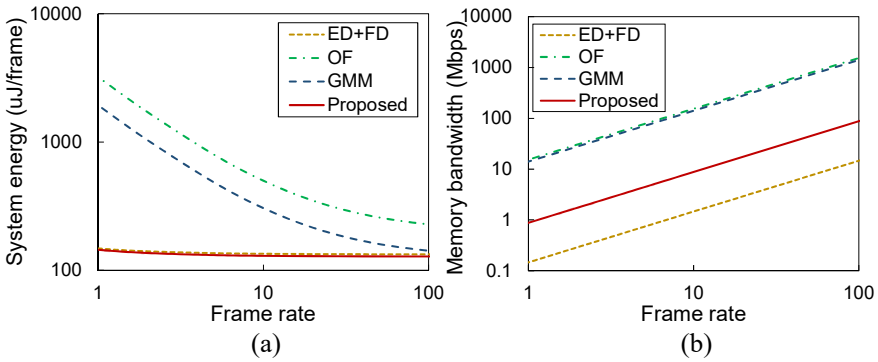


Figure 4.13: (a) System energy for ROI quality (SSIM)=0.8 with snowfall and (b) memory bandwidth with varying frame rate.

4.1.6 Discussions

Effect of Frame Rate

Since the memory leakage energy per frame increases with longer frame interval, the system energy saving of the proposed approach becomes more significant as a frame rate decreases [Figure 4.13(a)]. This result indicates that the proposed approach can be more attractive for the applications with low frame-rate requirements.

Effect on Memory Bandwidth

Using off-chip non-volatile memory to store the moving object detection parameters can potentially reduce the chip energy for GMM and OF based platforms; but the memory bandwidth requirements of the algorithms will finally determine the systems performance and energy with off-chip memory. Note off-chip memory will require higher access latency and energy. Due to need to read and update a large number of parameters, memory bandwidth requirements of OF and GMM are 210.4 and 191.9 bit/pixel, respectively, while the proposed method require only 12-bit access per pixel. Figure 4.13(b) shows that GMM and the proposed approach requires 442Mb/s and 4.6Mb/s, respectively, at 30 frame/s. Hence,

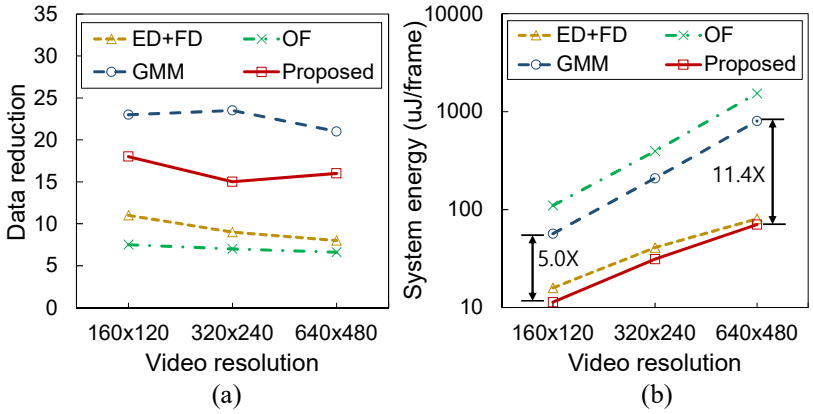


Figure 4.14: (a) Data reduction and (b) system energy consumption with varying video resolution for ROI delivery=0.8 with snowfall video.

even with off-chip memory, the proposed system is expected to consume less energy and provide better performance.

Effect of Video Resolution

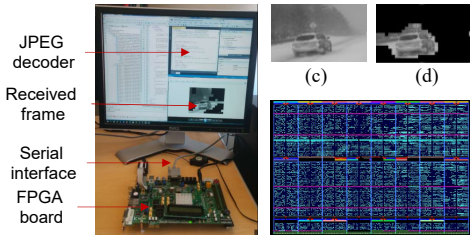
Video resolution can also affect the performance of the block-wise ROI decision scheme. Figure 4.14 analyzes the data reduction and system energy of different methods when ROI delivery is 0.8 with various resolution of the snowfall video. It can be observed that, the data reduction with proposed approach remains similar for all resolutions. However, higher resolution yields more energy advantage for the proposed method (compared to GMM). This is because the system energy at lower resolution is more dominated by transmission energy; while memory area/energy increases linearly at higher resolution, especially for GMM/OF.

FPGA Demonstration

I synthesize the proposed system with the MJPEG and moving object detection unit into the Xillinx Virtex-5 FPGA. Figure 4.15(a) shows resource utilization and computation energy of the FPGA implementation. The proposed approach consumes only 0.21nJ/pixel for

Logic block	Entire System	ED+FD +closing
LUTs	16869	280
FFs	13979	104
BRAMs	5	0
SRLs	22	0
DSPs	64	0
Energy (nJ/pixel)	**15.2**	**0.21 (1.4%)**

(a)



(b)

(c) (d)

(e)

Figure 4.15: (a) Resource utilization and computation energy, (b) Test environment, (c) original video frame, (d) received video frame, and (e) layout of the FPGA implementation.

computation, which is 1.4% of computation energy of the entire system. Figure 4.15(b) shows the system demonstration framework, where we supply the original frame image [Figure 4.15(c)] and reconstruct received frame image [Figure 4.15(d)] using a JPEG decoder software. Figure 4.15(e) shows the layout of the synthesized system.

4.1.7 Summary of the Section

This section presented a noise-robust moving object detection method based on simple operations that require low memory and computation. Its improved robustness to the environmental noise reduces transmission energy, enhancing the resource efficiency of the system with a reliable delivery of ROI. Therefore, the proposed design can be a lightweight alternative to complex moving object detection methods for resource-constrained IoT sensors. Also, the system-wide comparative analysis in this work demonstrates the need for an energy-aware algorithmic innovation in IoT design practices for achieving the challenging goal of improving quality under tight resource constraints.

4.2 Energy-Efficient ROI-Based Coding

4.2.1 Introduction

In the previous section, I presented a low-power noise-robust ROI detection approach. Once the ROI is detected, the ROI and non-ROI can be discriminated by an ROI-based coding method so that the system achieve higher energy efficiency for ROI delivery. The simplest approach to ROI-based coding is to drop non-ROI blocks and encode/transmit only the ROI blocks [20], as assumed in the previous sections. This approach reduces encoder energy and overall data volume as non-ROI blocks are not encoded or transmitted. However, it can suffer from the loss of context information and the ROI quality degrades significantly in case of the false negatives. To address these drawbacks, the non-ROIs should be delivered, with reduced resource and quality.

In this section, I present a tunable and low-complexity ROI-based coding scheme. The proposed approach is demonstrated on a wireless video sensor platform for moving object surveillance. A lightweight algorithm based on edge detection and frame differencing is used to detect ROI blocks. The number of ROI blocks is controlled by a threshold value. To allocate more data to the ROI, ROI and non-ROI blocks are encoded at the different data rates by compressing the non-ROI more. For more compression of the non-ROI, we pre-process non-ROI blocks using bit-truncation, which reduces the variance in pixel values in a block and allows higher compression for a given quality factor (QF) in MJPEG. Bit-truncation also reduces the energy dissipation of MJPEG.

The proposed video surveillance platform is designed as a fully synthesizable digital computation engine. The platform is simulated using 130nm CMOS for energy analysis, and prototyped into Virtex-5 FPGA for functional validation. The analysis shows that the non-ROI pre-processing using bit-truncation has lower energy consumption and better tunability compared to the prior works using the different QF values for the ROI/non-ROI [22] and pre-filtering of the non-ROI [23][24]. The sensor platform with the proposed

approach consumes 61% less system energy than H.264/AVC at the same ROI quality.

4.2.2 Proposed Approach

System Model

In this work I consider a block-wise ROI processing model, since it is compatible with the recent encoders that are based on block processing. Each 8x8 macroblocks (MBs) in a frame is determined as the ROI/non-ROI. Then, ROI MBs are directly encoded by MJPEG, while non-ROI MBs are pre-processed before encoding for higher compression.

Baseline ROI-detection engine

For a sensor platform design in this section, I define the ROI as the region with moving objects, which is detected by a low-overhead method based on edge detection and frame differencing (ED+FD) [11]. In the original ED+FD method, the edge maps of each MB for two subsequent frames are compared, and the MBs with the sum of the pixel-wise difference (i.e. activity level) larger than the threshold are determined as the ROI. To enhance its robustness to environmental noise, I improve the method by adding a morphological operation using rank-closing filters. However, I would like to note that the ROI-detection algorithm is not a key contribution of this section, and the proposed ROI-based coding approach is compatible with any ROI detection scheme.

Pre-processing of non-ROI MBs

For pre-processing of non-ROI MBs, I propose a tunable bit-truncation method to reduce both computation energy and the encoding rate of the non-ROI. The number of truncated bits serves as the control knob to reduce energy and encoded data volume at the expense of the quality degradation of the non-ROI. The bit-truncation method is a popular signal processing technique [86], however, tunable bit-truncation has not been applied before as a pre-processing technique for non-ROI blocks.

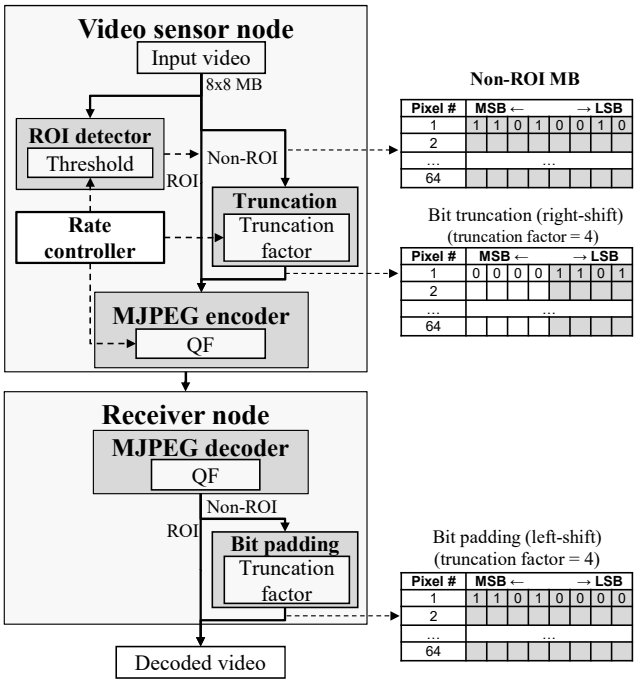


Figure 4.16: Diagram of the proposed bit-truncation method.

Figure 4.16 shows the overall procedure of the proposed approach. After the ROI decision unit classifies ROI/non-ROI MBs, low-order bits in 64 pixels of non-ROI MBs are truncated and encoded by the MJPEG encoder, while ROI MBs are encoded without truncation. After being transmitted to the receiver node, bits in the truncated non-ROI MBs are left-shifted by the same number of bits as in the truncation operation at the sensor node. The hardware design of the truncation method is simple because it is based on a bit-shift operation. Also, the truncation level can be easily tuned by controlling the number of shift operations. Therefore, the bit-truncation method can be effectively integrated with an on-line rate controller.

Comparison with Alternative Approaches

Figure 2.3 estimates the transmission overhead of the bit-truncation based approach. As the on-line rate controller can vary the number of low-order bits to truncate (truncation factor),

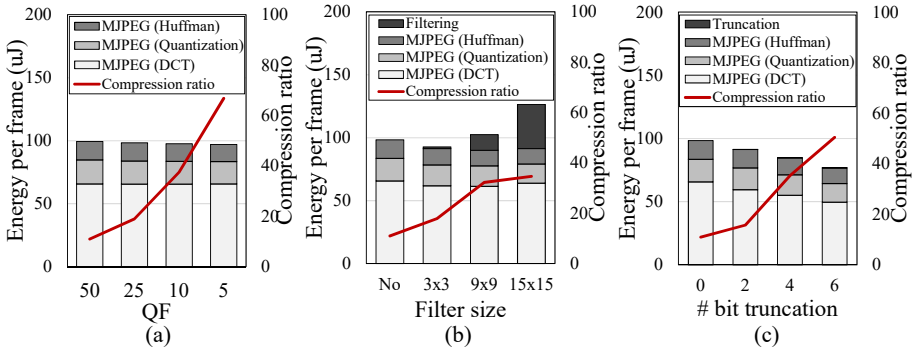
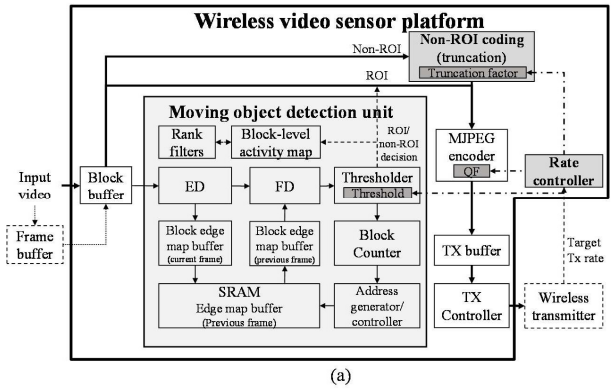
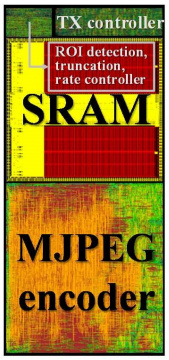


Figure 4.17: MJPEG computation power and encoded data size with variable parameter for (a) multi-QF, (b) pre-filtering, and (c) proposed approach.

this value should be included in the packet header together with the ROI map information [Figure 2.3(d)]. Although it incurs transmission overhead of 150.4 byte/frame (for 320x240 video), it is smaller than the multi-rate (1534 byte/frame) and the multi-QF method (151 byte/frame).

Figure 4.17 shows compression ratio and computation energy, which includes the MJPEG encoding and non-ROI processing (pre-filtering/truncation) energy, with varying parameters in each ROI-based coding method. At the same lowest compression, the multi-QF consumes more energy than the other two methods because it generates one more quantization table for the additional QF value. With higher compression in the multi-QF and pre-filtering methods, quantization and Huffman encoding energy slightly decreases because higher compression reduces non-zero values involved in the computation of these coding units [2]. However, the Discrete Cosine Transform (DCT) energy is not reduced since the input pixel values to the DCT do not decrease. Also, in the pre-filtering approach, computation energy increases with larger filter size as a larger filter needs to process more pixels. On the other hand, in the proposed bit-truncation method, energy consumption of the DCT decreases as more non-zero bits are truncated. Moreover, it reduces the variance of pixel values, resulting in energy saving in Quantization and Huffman encoding. The reduced variation also enables higher compression ratio at the MJPEG encoder, thereby reducing the non-ROI data size. Overall, more bit- truncation reduces data volume as well





Area and Energy Analysis in 130nm CMOS			
Module		Area (mm^2)	Energy (uJ/frame)
Moving Object Detection unit	Computation unit	0.021	0.55
	Memory (10 KB SRAM)	0.49	1.65
Proposed approach	Truncation unit	0.00024	0.0051
	Rate controller	0.03	2.8
	Total	0.03(1.1%)	2.8(1.9%)
MJPEG (variable QF†)		1.41	117.8
Tx controller		0.064	23.9
Tx buffer (12 KB SRAM)‡		0.61	2.11
Total		**2.63**	**148.8**

†Fixed-QF MJPEG: 1.02 mm^2, 68.7 uJ/frame
‡Tx buffer based on Flip-flops: 6.23 mm^2, 14.9 uJ/frame

(c)

Figure 4.18: The hardware implementation: (a) block diagram of the wireless video sensor platform, (b) the ASIC implementation (layout) of the platform, and (c) area and energy analysis (130nm CMOS).

as computation energy, at the expense of quality degradation. Also, the hardware cost of bit-truncation is much smaller than filtering and multi-QF encoding. Hence, the number of truncated bits can be used as an efficient control knob for an on-line rate controller.

4.2.3 Experimental Results

Full-chip Design

A wireless sensor node with the proposed approach [Figure 4.18(a)] is implemented in ASIC with 130nm CMOS [Figure 4.18(b)]. The ASIC implementation operates at 128 MHz (1.2V), supports 40 frames/sec, and consumes processing energy of 148.8 uJ/frame. The proposed ROI-based coding unit and the rate controller occupy only 1.1% of area and 1.9% of energy of the entire processing engine [Figure 4.18(c)].

Simulation Framework

For the system-level energy analysis, the energy associated with processing and memory is obtained from the simulation results of the ASIC implementation. Transmission energy is modeled from a wireless transmitter by Nordic Semiconductor (nRF24L01+) [73], which consumes 23.1 mW at 2Mbps of bandwidth and -18dB of signal power. For the per-

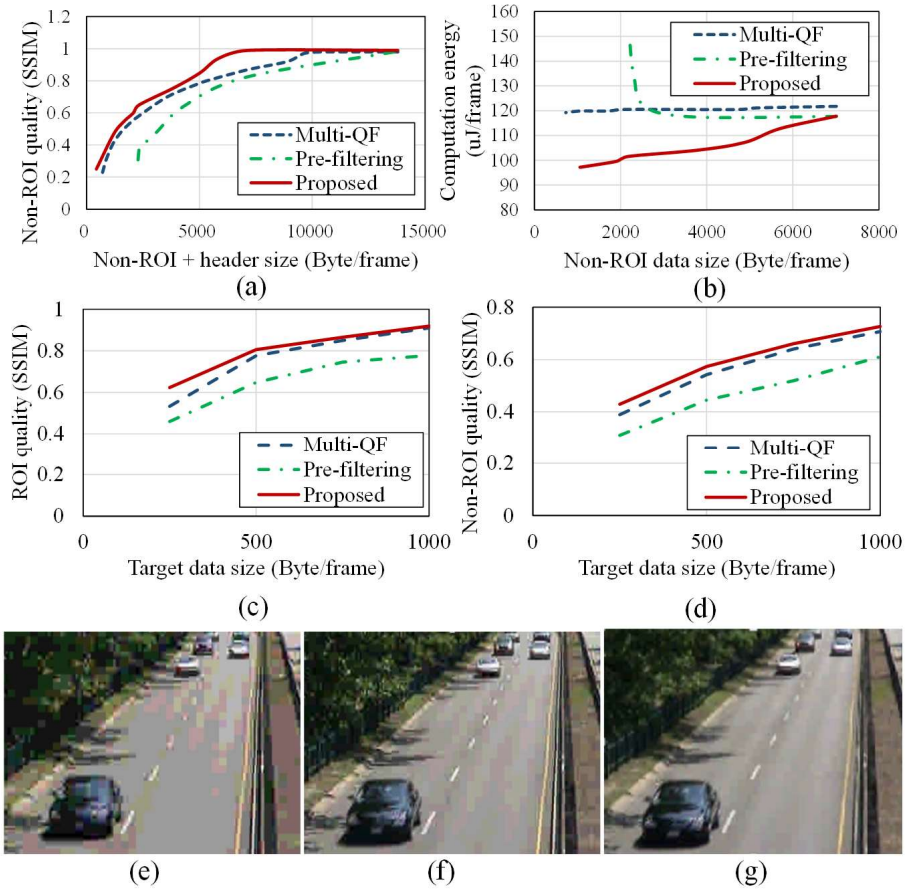


Figure 4.19: (a) Rate-quality and (b) computation energy for non-ROI. (c) ROI and (d) non-ROI quality vs. target size with the rate controller. Images at target data size=500B/frame with (e) pre-filtering, (f) multi-QF, and (g) proposed approach.

formance comparison of the ROI-based coding methods, I develop a MATLAB/Simulink-based simulation framework that includes the proposed method, multi-QF, and pre-filtering with a Gaussian filter. For H.264/AVC, the quality and rate-control performances are simulated through the JM 18.6 reference SW [74], and energy consumption values are scaled to 130nm from the existing study [87]. I use two video sequences from the 2014 dataset in changedetection.net [88]; highway and snowfall. I assume 320x240 resolution and a 10 frames/sec frame rate.

Comparison of the ROI-based coding schemes

To compare the rate-quality performance of the different ROI-based coding approaches, I simulate the data size and quality of the non-ROI by varying the parameters of each approach. Figure 4.19(a) indicates that the proposed approach shows the highest non-ROI quality under a given data size of the non-ROI with the MJPEG header. Also, Figure 4.19(b) shows that the proposed method has less computation energy than the other two methods due to reduced MJPEG energy and simple bit-shift operations. I compare the effectiveness of three coding methods under a given total data rate constraint. I use the same three-parameter rate-controller scheme as in Figure 6; the third PID controller controls the low QF and the filter size for the multi-QF and the pre-filtering methods, respectively. Because of the better rate-quality performance on the non-ROI, the proposed approach provides higher ROI quality under a given target data rate [Figure 4.19(c-g)].

4.2.4 Summary of the Section

This section presented an energy-efficient parameter-controllable ROI-based coding scheme that enhances the ROI quality with lower system energy consumption. Its better rate-quality characteristic for non-ROI enables the system to allocate more bits to the ROI, thereby providing higher ROI quality. Its low-complexity hardware also reduces computation energy. The PID-based rate controller minimizes buffer size by dynamically tuning the coding parameters for the constant encoding rate. With its improved energy-efficiency, the proposed approach can be an attractive solution for reliable delivery of visual information in resource-constrained video sensor nodes for surveillance applications.

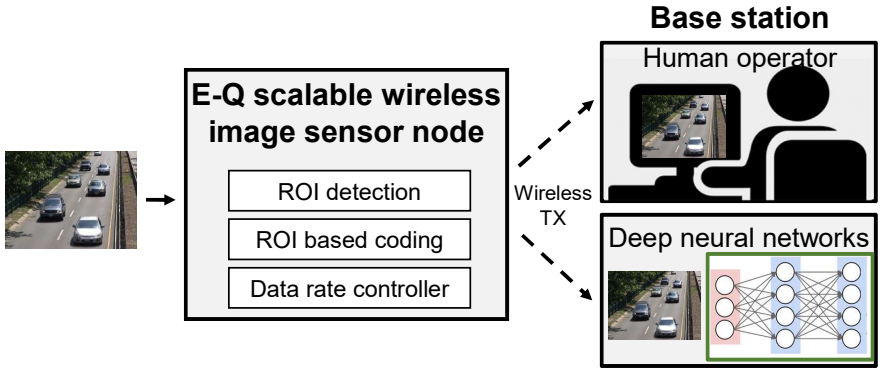


Figure 4.20: Configuration of an intelligent surveillance system with a wireless sensor node and a base station. We assume the base station with either human operators or deep neural networks monitoring objects.

4.3 Effect of ROI-Based Processing on Neural Network Inference

4.3.1 Introduction

In preceding sections, I assumed conventional image sensor systems that focus on the capabilities of wireless sensor nodes to capture, process, and transmit visual information to the base station, while leaving the task of video analysis to human operators [1]. In this configuration, delivering images with higher perceptual quality to human operators is critical. Therefore, the preceding sections explored resource-quality scalability to optimally allocate limited resources for better visual information.

However, the major drawback of the human operator based surveillance system is that the human monitoring of video is extremely labor-intensive and highly prone to error [89]. To reduce or eliminate the need for human involvement, automatic video analysis technologies have been adopted to build intelligent surveillance systems [90]. With recent success of deep neural networks on various vision tasks [91][92], deep-learning based monitoring systems are gaining more attention [93] [Figure 4.20]. In such a system configuration, the most critical performance metric is detection/classification accuracy of the neural network.

Therefore, the image sensor node should be designed to process the video in a way that the neural network achieves higher accuracy. Conventionally, ROI-based processing in a sensor node is targeted to provide better perceptual quality of the ROI, but better ROI quality may not be directly translated into higher classification performance of neural networks. Several studies have shown that image quality distortions to human perception impose different impact on classification performance of neural networks [33][32]. However, all the studies assumed uniform degradation of the images due to the unwanted distortion. As ROI-based processing is a critical component of wireless image sensor nodes for energy optimization, it is critical to analyze the effect of various ROI-based processing approaches on the scalability of the energy and classification accuracy. This section presents the design of an ROI-based energy-quality scalable wireless image sensor node with focus on an intelligent surveillance system based on a deep-learning-based object classifier. By designing an intelligent surveillance system using two types of object classifiers with deep neural networks, I investigate how the energy-quality scalability of ROI-based processing is translated into the energy-accuracy scalability. Better energy-quality scalability enabled by the proposed approach leads to higher energy-accuracy scalability for the base system with neural networks. I also show that the impact of the ROI/non-ROI quality on the neural network accuracy depends on the classification type of the network.

4.3.2 Surveillance System Framework

To build an end-to-end surveillance system, I consider two types of a video monitoring scheme [Figure 4.21]; a base station with human operators and the one with deep neural networks. In the first configuration, human operators are monitoring the video frames transmitted by an image sensor node, focusing on the behavior of moving objects. As the human monitoring performance is highly dependent on the perceptual quality of the ROI, providing better quality scalability under available energy is critical. In Section V, I analyze the energy-quality scalability of the ROI-based processing algorithms. I also consider a

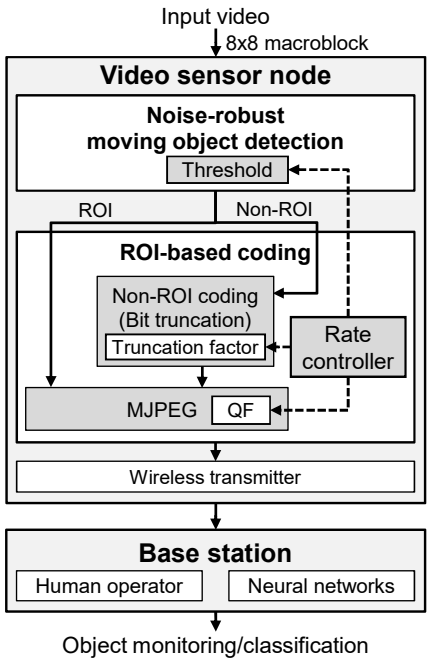


Figure 4.21: Block diagram of the surveillance system with a wireless sensor node and a base station.

host system that performs automatic object classification using deep neural networks. I assume the neural network is deployed after being trained for image frames with target objects and corresponding class labels. Then each frame image received from a sensor node is supplied into the neural network for an inference of the object class. Generally, the object classification using deep neural networks can be divided into two types; frame-based classification and region-based classification. Frame-based classification identifies salient object classes in the whole frame image. On the other hand, region-based classification first localizes the objects in a frame image, and then classifies the object in each location. For both neural network classifiers, the critical performance metric is object classification accuracy rather than the perceptual quality. Therefore, sensor nodes are required to process the video so that the neural network achieves higher classification accuracy under a given energy constraint. Section VI analyzes the energy-accuracy scalability of the ROI-based processing algorithms depending on the type of the classification network.

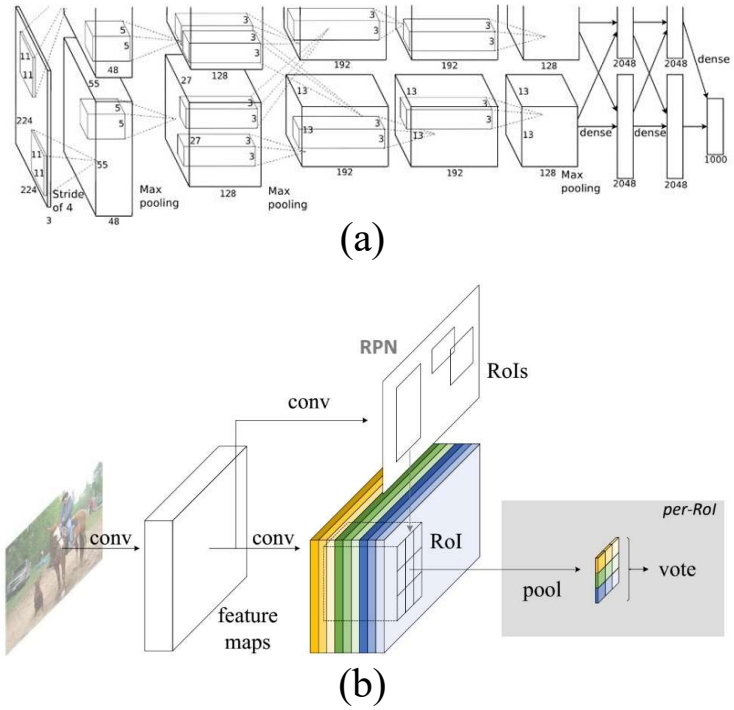


Figure 4.22: Structure of the neural networks for experiments. (a) AlexNet [91] and (b) R-FCN [94].

4.3.3 Experimental framework

In this section, I analyze the effect of ROI-based processing approaches on the classification accuracy of the neural network. I focus on how the energy-quality scalability is translated into energy-accuracy scalability for the two types of the classification schemes. Here, I assume the moving objects are the objects that I want to classify using neural networks.

For experiments, I use AlexNet [91] for frame-based classification and R-FCN [94] for region-based classification [Figure 4.22]. AlexNet is a commonly-used neural network for image classification, having five convolutional layers and three fully-connected layers. I use the Caffe framework [95] with the reference model trained for the ImageNet dataset [96]. R-FCN is a region-based, fully convolutional network designed for object localization and classification. It first proposes class-agnostic bounding boxes using the region proposal network. Each bounding box is then supplied to the backbone neural network

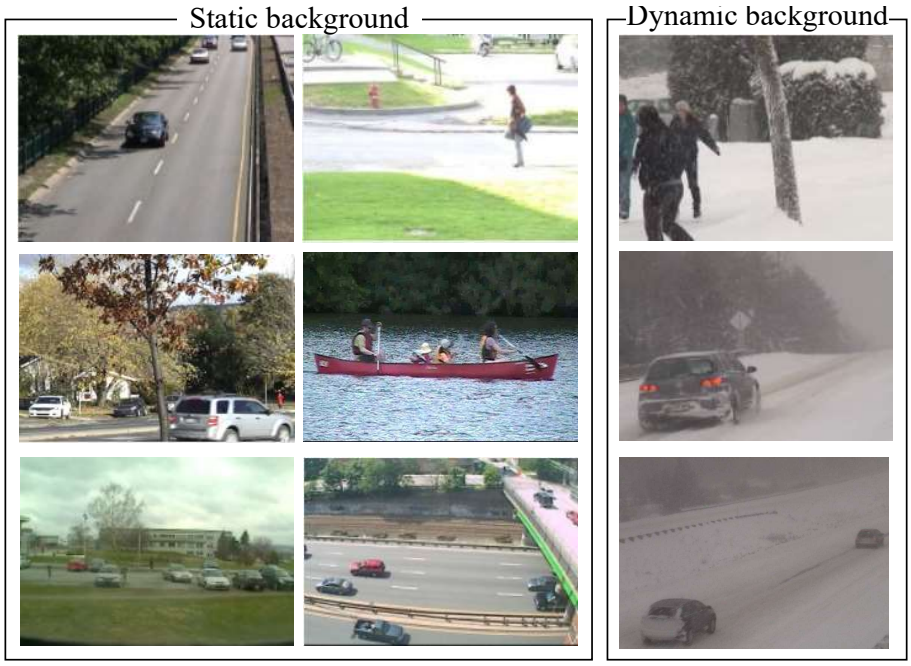


Figure 4.23: Frame images of test video sequences. (Top from the left) highway, pedestrian, and skating. (Middle from the left) canoe, fall, and snowfall. (Bottom from the left) parking, streetlight, blizzard.

for classification. The final output of the network is a list of bounding boxes with class probability higher than the pre-defined confidence threshold (default=0.5). I use Python version of the R-FCN implementation [97] pretrained for the MS COCO dataset [98]. For the backbone image classification network, I use ResNet-101 model [99].

Test video sequences used in this section are highway, fall, parking, streetlight, canoe, blizzard, and snowfall [Figure 4.23]. Two videos used in Section V (pedestrian, skating) were replaced with fall, parking, streetlight, blizzard, and canoe since main objects in pedestrian and skating videos are the person, which is not a part of ImageNet classes. For accuracy metric, I use top-1 accuracy for the classification network and precision/recall/accuracy for the object detection network, as generally used in the field. The reason why I used different metrics for different network types is that, the classification network generates only one list of confidence values of classes, while the object detection network detects multiple locations with corresponding object confidence level. Therefore,

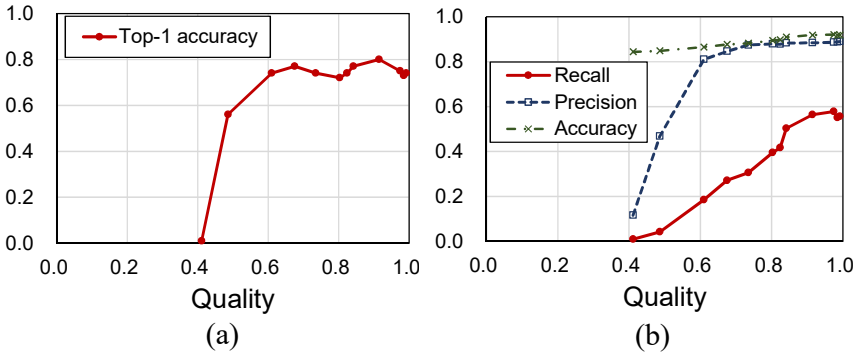


Figure 4.24: (a) Top-1 accuracy of AlexNet and (b) accuracy metrics of R-FCN for varying quality of highway video.

analysis of frame-based classification uses the top-1 accuracy, which is the portion of the images that correct class matches the highest-probability class predicted by the neural network. In classification of video frames, I define the top-1 accuracy as the portion of the frames where the class of highest probability belongs to the set of correct classes. In region-based classification, the classification performance metrics (accuracy, precision, and recall) are computed based on the areas of ground-truth bounding boxes and the ones generated by the network for each object class. The area of the generated bounding boxes is multiplied with the probability associated with each bounding box.

4.3.4 Analysis Results

Effect of Uniform Quality Degradation

It is known that encoding process of JPEG compression such as quantization results in loss of information and quality degradation [100]. I first examine how the quality degradation due to JPEG encoding affects the classification performance. In this experiment, the quality of the highway video frames is uniformly degraded by varying the single QF of the JPEG encoder. Figure 4.24(a) shows the relation between the quality and the top-1 classification accuracy of AlexNet. The figure shows that the relationship is not linear; the accuracy remains high when the quality is higher than certain point, but it shows steep decrease if the

(a) (b)

Figure 4.25: Output frame image of R-FCN when (c) quality = 0.4 and (d) quality = 0.95

quality decreases beyond that point. This result indicates that there is certain threshold of quality that the neural networks can extract features required to perform correct classification.

In region-based classification, recall decreases almost linearly as the quality degrades [Figure 4.24(b)]. As I degrade the frame quality, classification on each bounding box will have lower probability for the correct object class. Therefore, for a certain confidence threshold there will be less bounding boxes, resulting in lower recall [Figure 4.25(a-b)]. On the other hand, precision shows similar trend as the frame-based classification accuracy. This is because the quality degradation of the bounding box areas affects the classification performance of the backbone neural network similarly as in the frame-based classification.

Figure 4.26 shows the classification performance of different video sequences. For frame-based classification with AlexNet, all the videos show non-linear relationship between the quality and accuracy. It can be also observed that the baseline accuracy (when quality=1) varies a lot across the video sequences. This variation comes from the difference in the target object and the background of the videos. For example, while the four video sequences (highway, streetlight, fall, snowfall) have the same target object classes, the highway and streetlight videos show higher baseline accuracy than the other two videos because the car objects are moving in the usual background (road). When cars are moving

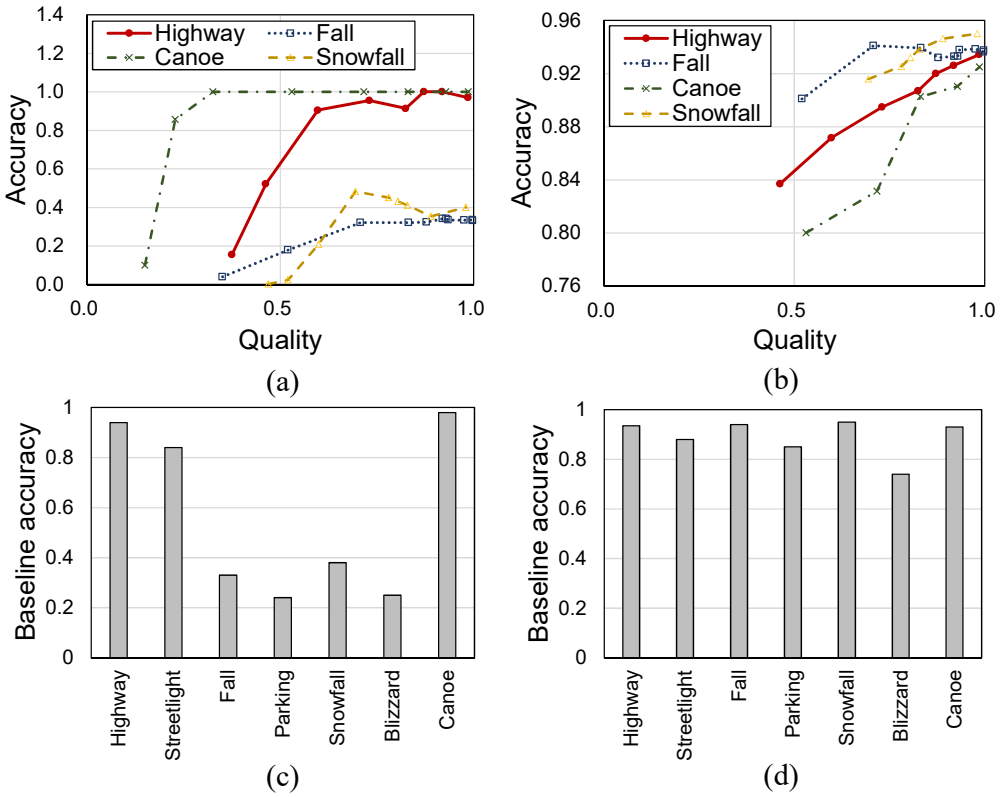


Figure 4.26: Quality-accuracy scalability of (a) AlexNet and (b) R-FCN for different video sequences. Baseline accuracy with the original images (quality = 1.0) for (c) AlexNet and (d) R-FCN.

in an unusual background with trees (fall) or buildings (parking), it shows lower accuracy. If the video has environmental noise (snowfall, blizzard), the neural networks tend to show lower baseline accuracy than clean videos because the noise (e.g., snow particles) corrupts the object feature. On the other hand, in region-based classification using R-FCN, the baseline accuracy is similar across the video because the accuracy is dependent on the resolution of the objects, not the quality or context of the background. Figure 4.26 shows that the videos with relatively smaller objects (parking, streetlight) have lower baseline accuracy than the videos with larger objects (highway, fall).

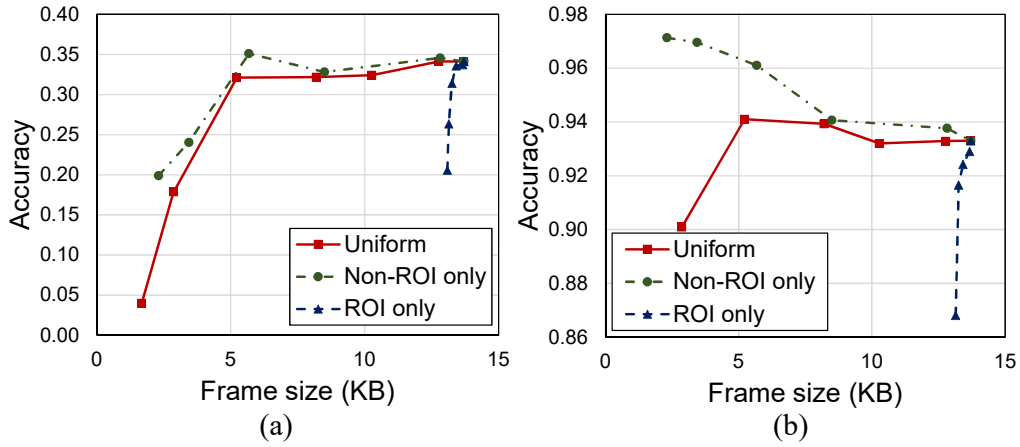


Figure 4.27: Accuracy for the average frame size of the frame when uniform/non-ROI only/ROI only degradation of the frames with (a) AlexNet and (b) R-FCN.

Effect of ROI/Non-ROI Quality Degradation

In this experiment, I compare the uniform degradation with the case where the ROI and non-ROI area are differently degraded. For example, a video with non-ROI only degradation was generated by decreasing the QF of the non-ROI blocks from 50 to 3, while keeping the QF of the ROI blocks as 50. Figure 4.27 shows that in both types of classification, reducing the ROI quality significantly affect the classification performance while the non-ROI quality has less impact on the accuracy. Especially in region-based classification, the non-ROI quality degradation does not even decrease the classification accuracy because the region proposal and classification performance depends only on the quality of the ROI. Also, when the quality of the background is degraded, the network is less likely to detect object at the background, which is the false positive.

Figure 4.28 shows the quality of the frame required to achieve the target classification performance. The target accuracy is set to 50% and 90% of the baseline accuracy (accuracy at the quality=1.0) for the frame-based and region-based classification, respectively. For all the videos, the quality can be lower when degrading only the non-ROI quality than degrading the ROI quality, for the same classification performance. I note that the difference of

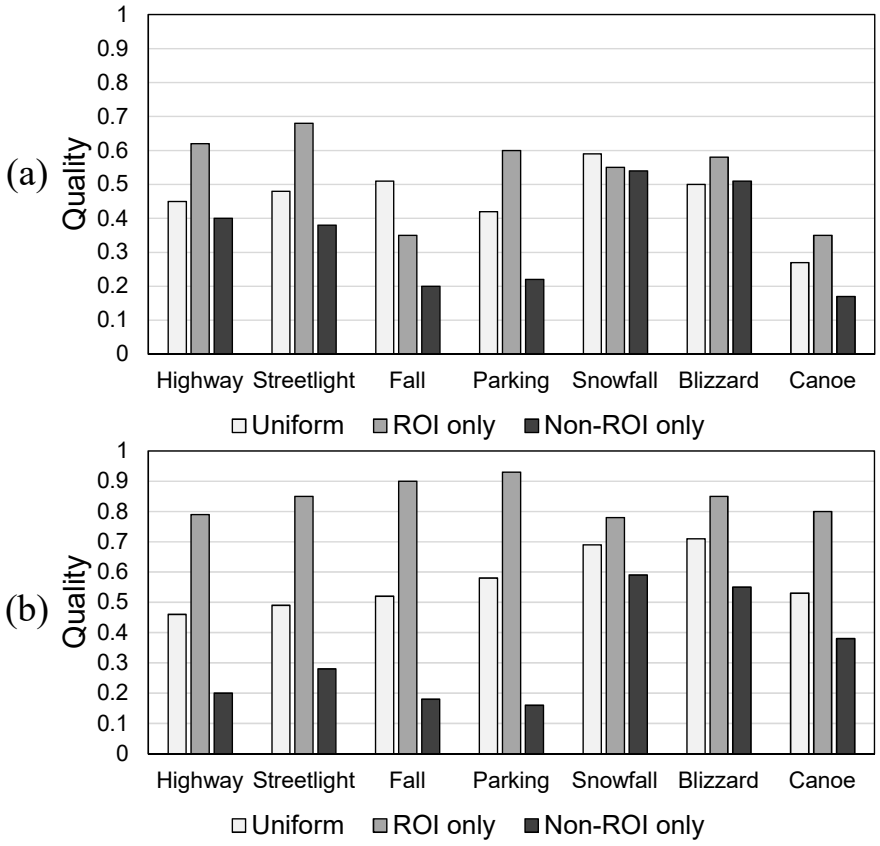


Figure 4.28: Required quality for certain target classification accuracy. (a) AlexNet (target = 50% of the baseline accuracy) and (b) R-FCN (target = 90% of the baseline accuracy).

the required quality between the ROI/non-ROI degradation is higher in the region-based classification, than in the frame-level classification. Based on these results, I can justify the use of ROI coding that for the neural-network based monitoring system. It can be also observed that for noisy videos, further distortion due to encoding magnifies the effect of accuracy degradation. Therefore, compared to the clean videos, noisy videos generally require higher quality to satisfy the same level of accuracy loss. Also, the noisy videos require higher quality of non-ROI than clean videos since non-ROI encoding makes the objects of interest less distinct from background, leading to lower accuracy. For the videos with unusual background (fall, snowfall), reducing non-ROI quality does not affect the accuracy much because the background information is not important for accuracy classification. Also, as streetlight and parking videos have smaller object size, it becomes more

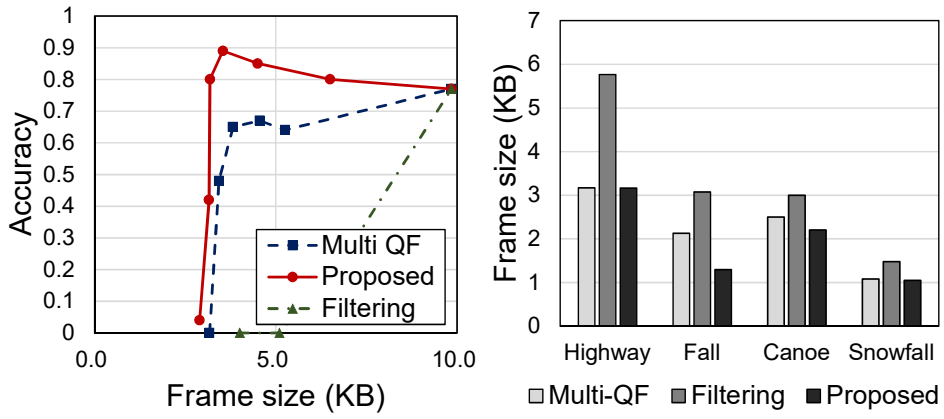


Figure 4.29: (a) Frame size vs. accuracy of AlexNet classification for different ROI coding methods. (b) Required frame size to achieve a target classification accuracy (50% of the baseline accuracy).

important to keep high quality of ROI to preserve information for the target inference accuracy.

Effect of ROI coding scheme

In the previous section I observed the non-ROI quality has less impact on the classification accuracy. Now I analyze how we need to reduce the non-ROI size and quality to maximize the accuracy under target frame size constraint. For each of the ROI coding schemes, video frames are generated by changing the ROI coding parameters while keeping the QF of the ROI area to be 50. Here, I assume the ground-truth ROI information is given to the ROI coding methods. I do not show the comparison on the region-based classification because reducing the non-ROI quality has little impact on the accuracy of the region-based classification, as mentioned in the previous section.

As shown in Figure 13 in Section V, the proposed ROI coding method provides higher non-ROI quality for a given data rate with its ability to restore information by rescaling (shifting bits) and smaller header overhead. Better rate-quality scalability of the proposed

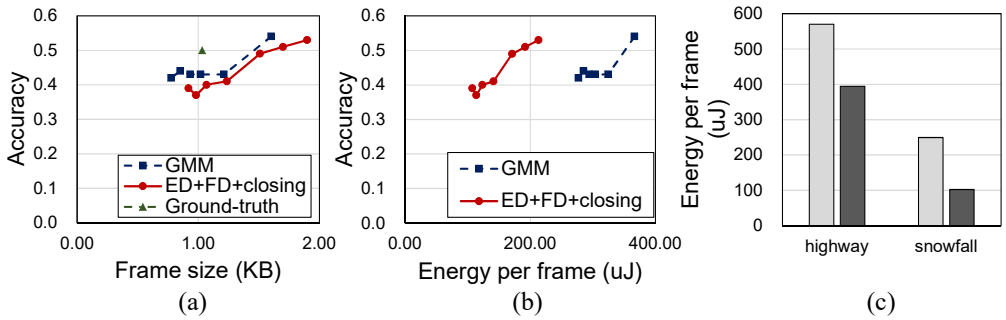


Figure 4.30: (a) AlexNet accuracy for varying (a) frame size and (b) energy per frame with various moving object detection methods. (c) energy consumption per frame for a target classification accuracy.

ROI coding method is translated into better rate-accuracy scalability, as illustrated in Figure 4.29(a). Figure 4.29(b) shows the required frame size to achieve the target classification accuracy (50% of the baseline accuracy). For all the videos, the proposed method can reduce the non-ROI size more than other methods for the target accuracy.

Effect of Noise-Robust Moving Object Detection

When the ROI detection algorithm falsely identifies the ROI as the non-ROI, the region is subject to degradation due to the ROI coding method. This will result in lower quality of the ROI, which will also degrade the classification performance. Therefore, the classification accuracy will be also dependent on the ROI detection performance.

In this experiment, I assume the video frames are encoded by the proposed ROI coding method based on the ROI information obtained by the proposed noise-robust detection method (ED+FD+rank closing) and GMM. Figure 4.30(a) shows that using two moving object detection methods leads to lower accuracy than using the ground-truth ROI information because of their false detections. GMM shows better classification performance than the proposed detection method since it performs better ROI detection as shown in section V. However, the GMM requires much larger energy to achieve the same level of detection accuracy. Therefore, as Figure 4.30(b) shows, the proposed method shows much better energy-accuracy scalability than GMM. This leads to 1.5 to 2.4X lower energy consump-

tion for the target accuracy, as illustrated in Figure 4.30(c).

4.3.5 Summary of the Section

This section presented an ROI-based energy-quality scalable wireless image sensor node for intelligent surveillance systems. Improved noise robustness of the ROI detection method reduces transmission energy with a reliable delivery of ROI. The low-overhead ROI coding method enables better rate-quality scalability, and a simple rate controller minimizes buffer size by dynamically tuning the parameters. Improved energy-efficiency of the proposed approach leads to better energy-accuracy scalability for the intelligent surveillance system using deep-learning based object classification.

CHAPTER 5
LOW-POWER DATA RATE CONTROL FOR A MEMORY-EFFICIENT IMAGE SENSOR SYSTEM

5.1 Energy-Aware Control of Target Data Rate

5.1.1 Introduction

The optimization of video quality and energy consumption also involves variation in a wireless channel condition. Since a noisy channel condition increases distortion of a transmitted video due to bit errors, a transmitter usually enhances channel reliability at the cost of a lower channel data rate or higher signal power. However, this adaptation causes an increase in transmission energy or degradation of the quality (dropping of frames) resulting from reduced available bandwidth. Therefore, achieving an optimal system-level energy-quality tradeoff under varying channel conditions requires system-wide feedback control that adaptively tunes parameters of an encoder and a transmitter.

5.1.2 Proposed Control Method

In many wireless systems, the transmitter is able to estimate a wireless channel condition and adapt to the condition by changing its operation mode [41]. The proposed algorithm assumes that the transmitter has a set of discrete operation modes, each consisting of a pair of parameters transmission signal power (P_{TX}), channel rate (R_C). For a fixed RC, increasing P_{TX} will increase the received signal-to-noise ratio (SNR) and result in a lower BER. Also, for given P_{TX}, decreasing the RC will lead to a decrease in a BER because of reduced receiver sensitivity [101]. However, these changes for higher reliability result in

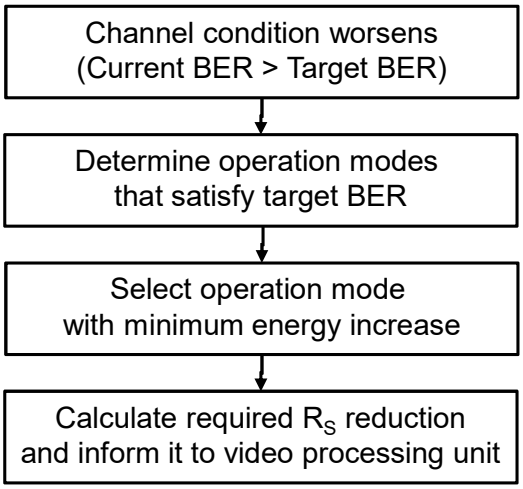


Figure 5.1: Procedure of the energy-aware control scheme

an increase in transmission energy, which is given by [102]

$$E_{TX} = (P_{Base} + P_{RF} + P_{TX}/\eta) \cdot (R_S T_{Fr})/R_C \tag{5.1}$$

where P_{Base} denotes power consumption of a baseband unit, P_{RF} denotes power consumption of an RF unit, η is drain efficiency, R_S is the source data rate, and T_{Fr} is a frame interval. As this equation shows, when transmission energy increases due to a P_{TX} increase or an R_C decrease, we can keep the energy constant by reducing the R_S accordingly. I design the energy-aware control scheme that controls the R_S for maintaining target energy consumption under operation mode changes due to the variation in channel conditions. Maintaining the target energy consumption is crucial in low-power systems since their applications usually require that they operate for a certain time with limited energy sources.

Figure 5.1 illustrates the procedure of the energy-aware control scheme. First, if the current mode does not satisfy the BER target because of decreased channel reliability, it determines the alternative operation modes that satisfy the BER target. When determining

Control scheme	Feedback	Awareness		Source rate reduction	Parameters for source rate control
		Energy	Content		
Independent control	No	No	No	No source rate control	
Energy and content unaware control	Yes	No	No	When $R_C < R_S$	Quality of all MBs
Energy-unaware content-aware control	Yes	No	Yes	When $R_C < R_S$	Quality or priority of each MB
Energy-aware content -unaware control	Yes	Yes	No	When $P_{TX} \uparrow, R_C \downarrow$	Quality of all MBs
(This work) Energy and content aware control	Yes	Yes	Yes	When $P_{TX} \uparrow, R_C \downarrow$	# MBs to encode, quality of encoded MBs

Table 5.1: Summary of control schemes

operation mode, I use a concept of BER target instead of a detailed BER-quality model since the error performance of the image quality can be modeled as a low pass filter. That is, while the low BER has almost no effect on the quality, quality drops off significantly after the BER increases past a certain level [103]. Second, the control scheme calculates transmission energy consumption for all the operation modes determined in the first stage and chooses the mode with the minimum increase in transmission energy. By selecting a mode with the minimum energy increase among the modes satisfying the BER target, it minimizes energy consumption with bounded quality distortion due to BER. This procedure improves energy-efficiency over the conventional algorithms, which select the highest transmission rate that satisfies the maximum BER requirement [41]. Finally, it calculates the required source data rate to maintain transmission energy and inform the rate to the video processing unit, so that the unit can control its output data rate.

The main idea of the proposed scheme is to reduce the R_S according to a P_{TX} increase or an R_C decrease, while the existing control schemes decrease the R_S only when it becomes larger than the R_C. The quality degradation of the ROI, due to the reduced R_S, can be minimized by using a content-aware control scheme. If the video processor is unaware of the relative importance of each MB, it controls the quality of all the MBs using the MJPEG quality factor (QF) (or quantization parameter in the H.264/AVC [104]) to achieve

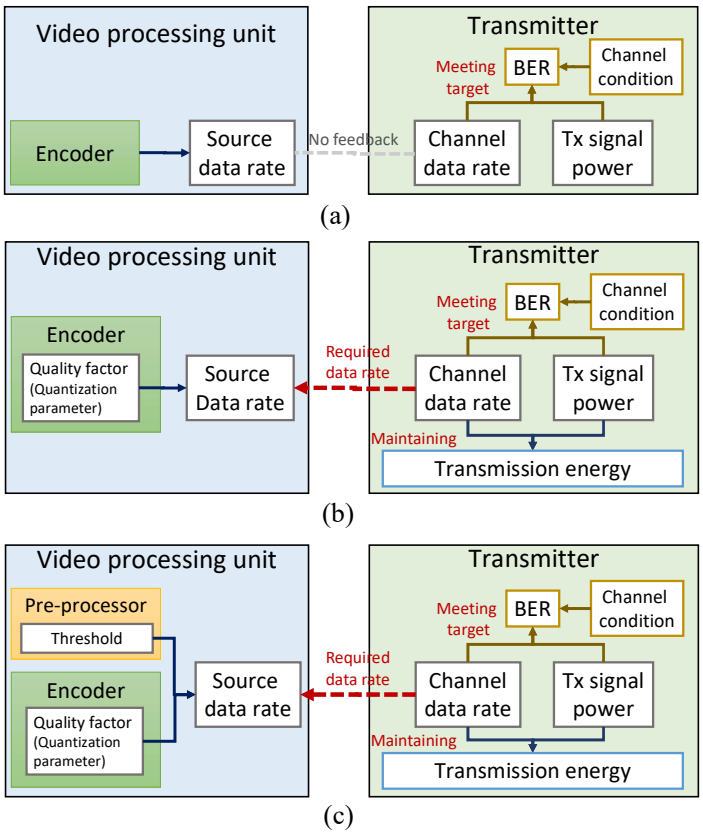


Figure 5.2: Diagram of control schemes. (a) Independent control scheme. (b) Content-unaware feedback control scheme. (c) Energy- and content-aware feedback control scheme.

the desired source data rate. The content-awareness enhances the energy-aware control since it better preserves the quality of the ROI at the same source data rate by dropping non-ROI MBs and increasing the quality of ROI MBs. The reduced number of encoded MBs leads to (i) a decrease in the source data rate (like other works), and (ii) reduction of computation energy at the encoder.

The characteristics of each control scheme are summarized in Table 5.1 and illustrated in Figure 5.2. By exploiting the awareness of both energy and content, the proposed scheme optimizes the system performance in two ways. First, the controller guarantees bounded transmission energy and quality distortion due to a wireless channel by reducing the source data rate according to the transmission parameters satisfying the BER target. Also, source-rate control using both the number of MBs to encode and the quality of these MBs further

Specification	VALUE
Frequency	2.4 GHz
Data rate (kbps)	250, 1024, 2048
Signal power (dBm)	0, -6, -12, -18
Receiver sensitivity (d Bm)	-94 @250 kbps -85 @1024 kbps -82 @2048 kbps
Transmission current consumption (mA)	11.3 @0 dBm 9.0 @-6 dBm 7.5 @-12 dBm 7.0 @-18 dBm

Table 5.2: Transmitter (nRF24L01+) specification

Channel loss (dB)		85	90	95	100
Tx signal power (dBm)		-6	0	0	0
Channel data rate (Kbps)		2048	2048	1024	250
Source rate (kbps) (Reduction)	Independent control, energy-unaware con trol	450 (5)	450 (5)	450 (5)	250 (9)
	Energy-aware control	450 (5)	360 (6.25)	180 (12.5)	45 (50)

Table 5.3: Transmission parameters and source data rate for channel condition

optimizes the quality of the ROI and the energy consumption in the computation.

5.1.3 Results

For system-level performance analysis, I use a Nordic Semiconductor nRF24L01+ transmitter (Table 5.2). The transmitter has twelve operation points, each with different channel rate and signal power (Figure 5.3).

I also analyze the performance of the control algorithms by varying channel conditions. As channel loss increases from 85 dB to 100 dB, the transmitter operates at lower band-

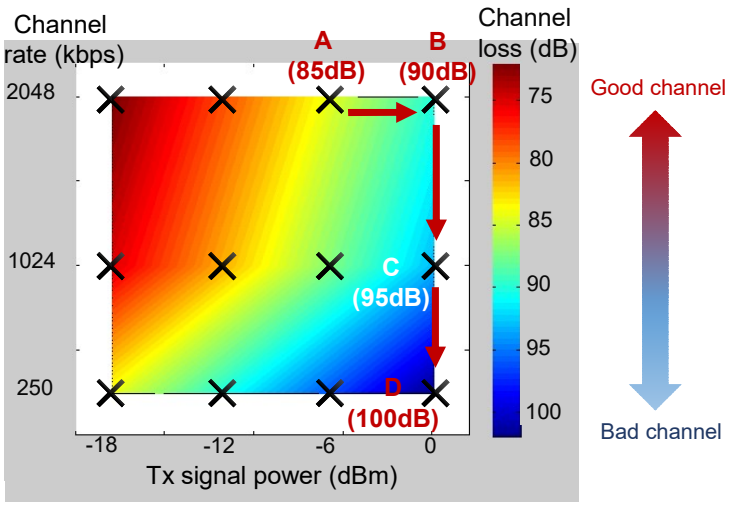


Figure 5.3: Operation modes of Nordicsemi nRF24L01+ and required channel loss for 10-6 BER.

width and higher signal pow-er to satisfy the BER target (ABCD in Figure 5.3). Table 5.3 shows the transmission parameters and data rate reduction for each channel condition and control scheme. Figure 5.4 shows that the independent controller and the energy-unaware controller significantly increase energy consumption because of a signal power increase or a channel rate decrease. The independent controller also shows a considerable quality degradation due to random packet drop at the wireless channel. However, the energy-aware control scheme maintains its energy consumption with little quality degradation by controlling the source data rate. The energy- and content-aware control scheme provides higher ROI quality than content-unaware control due to its better rate-quality characteristics. It also reduces energy consumption since it has reduced workload in the encoder.

5.1.4 Summary of the Section

This section presents a target data rate control mechanism for tolerance to variation in wireless channel, which is a vital requirement in the design of the resource-constrained sensor nodes for IoT. The proposed bounded-energy operation can enable sensor nodes in the IoT to perform more operations with energy constraints. Since wireless communication of the IoT is unreliable by nature, it is also important to adapt to wireless channel variation

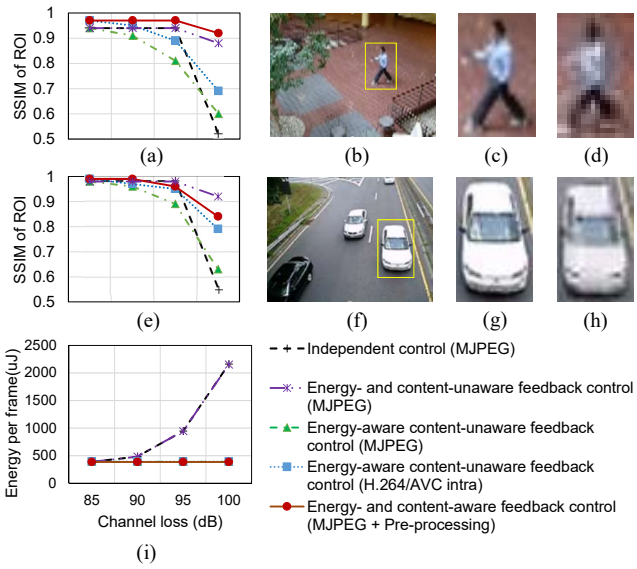


Figure 5.4: (a) Quality of ROI under varying channel condition for atrium video. (b) Original atrium video. (c) ROI image when energy- and content-aware feedback control (MJPEG + Pre-processing) used. (d) ROI image when energy-aware content-unaware feedback control (MJPEG) used. (e) Quali-ty of ROI under varying channel condition for traffic video. (f) Original traffic video. (g) ROI image when energy- and content-aware feedback control (MJPEG + Pre-processing) used. (h) ROI image when energy-aware content-unaware feedback control (MJPEG) used. (i) System energy consumption per frame under varying channel condition.

for reliable delivery of information. The proposed system-wide feedback control scheme preserves the quality of the ROI under variations in channel conditions.

5.2 Low-Power Control of Encoding Data Rate

5.2.1 Introduction

Chapter 4 examined that the proposed energy- and content-aware control scheme maximizes the quality of the ROI for a target data rate by determining three parameters (i.e., the threshold, QF, and quantization factor). In theory, it is possible to find an optimal choice for the two parameters that ensures the average source data rate for the entire video

stream satisfies the target rate while providing better ROI quality than alternative schemes, as shown in the rate-quality curves in Figure 3.9. However, an off-line analysis in which the parameters are statically determined based on the entire video stream is not a practical approach since an input video varies frame-by-frame, and the target source data rate also varies according to channel conditions. Therefore, to cope with these variations, the parameters should be controlled dynamically. In this section, I discuss the implementation of a dynamic source-rate controller that finds the threshold and the QF values on-line. I also describe a case study showing an empirical procedure for tuning coefficients of the controller, which can be improved for optimal coefficient tuning through future work.

In this section, I present a tunable and low-complexity ROI-based coding scheme with a multi-parameter on-line rate controller [Figure 2.4(b)]. The proposed approach is demonstrated on a wireless video sensor platform for moving object surveillance.

5.2.2 Challenges to the Data Rate Control

The size and the number of moving objects in a video stream usually vary by frame, resulting in variation in the activity-level distribution of MBs in a frame, as in Figure 5.5(a). Therefore, if the threshold value is fixed, the number of encoded MBs will change in each frame [Figure 5.5(b)]. When the QF is also fixed, this variation will lead to fluctuations in the source data rate [Figure 5.5(c)]. Even though the average source data rate for an entire video satisfies the channel rate, fluctuations in the source rate will cause a datarate mismatch between the encoder and a transmitter. The data rate mismatch will require a larger buffer between the encoder and a transmitter (energy/area overhead) to avoid randomly dropping data. For ex-ample, as Figure 5.5(c) shows, if the channel rate is 400 bytes/frame (32 kbps) and the threshold and the QF are fixed, the maximum amount of buffered data is 5,072 bytes, which is the size of a buffer required for the system. Therefore, to reduce the required buffer size, I propose a dynamic source-rate controller that regulates the parameters according to input video characteristics. In addition to reduction in the buffer

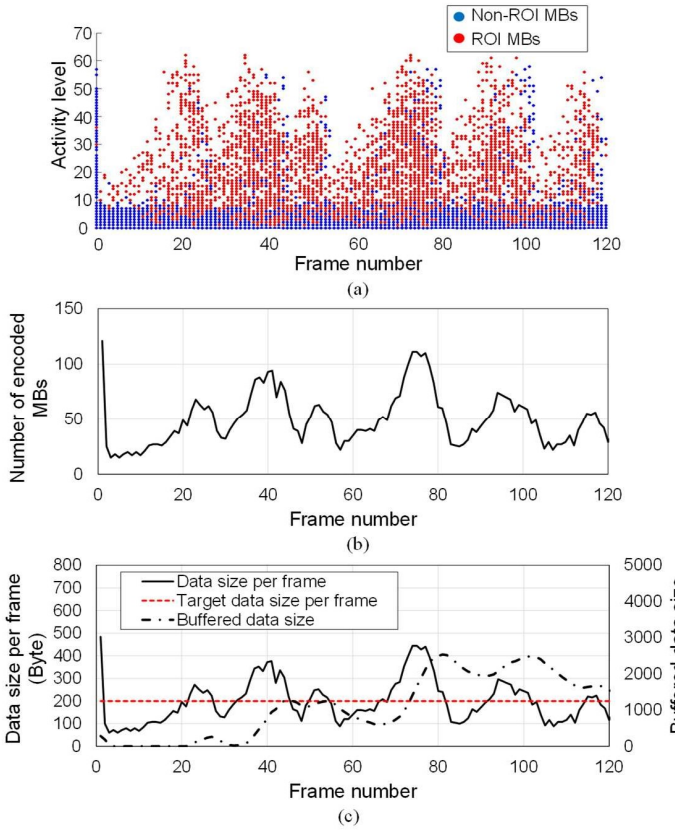


Figure 5.5: Simulation results with the fixed threshold (=5) and QF (=50) for the traffic video. (a) The distribution of the activity levels of MBs. (b) The number of encoded MBs. (c) The source data rate expressed as the data size per frame, and buffered data size.

size, regulating the source rate is also critical for the energy-aware control scheme, which aims to keep the target transmission energy constant by maintaining the source data rate. That is, even though the maximum transmission energy is limited by the channel rate, the source data rate that fluctuates below the channel rate will result in variation in transmission energy. Although the dynamic control of the parameters reduces the buffer size and transmission energy variation, it may yield lower quality of the ROI than the static determination of the parameters. For example, for frames with more ROI MBs than required to satisfy the target source rate, we should drop some of the MBs to satisfy the rate. However,

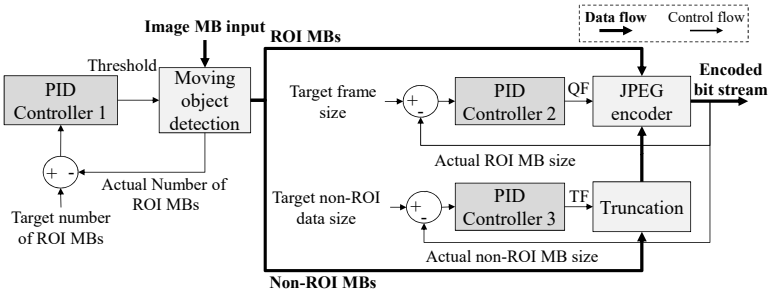


Figure 5.6: Diagram of the proposed on-line rate controller.

dropping more MBs with a higher threshold value may lead to a decrease in the delivery ratio of ROI MBs. Similarly, to keep the output data rate constant at the MJPEG, we should use a lower QF for a frame with more ROI MBs and a higher QF for a frame with fewer ROI MBs. Since more ROI MBs are encoded with a lower QF, the average quality of the ROI will degrade. In summary, the dynamic source-rate control of the content-aware processing scheme involves a tradeoff between the required buffer size and the quality of the ROI. A goal of the rate controller design is to dynamically change the parameters so that the source data rate satisfies the target rate and the quality of the ROI is maximized according to the required source data rate as well as the characteristics of the input video.

5.2.3 Proposed Data Rate Controller

To design a rate controller with less complexity, I adopt a proportional-integral-derivative (PID) controller. Since a PID controller is simple and applicable to a wide range of operating conditions, it is widely used in the automatic control area [105]. It is especially suitable for processes with unpredictability, one of the characteristics of the ROI-based video coding process in which the number of ROI MBs cannot be predicted for the coming frames. There are prior studies that used PID controllers in video coding to satisfy the target buffer fullness by modulating the source data rate [106][107]. However, our objective is distinct in the sense that I apply a PID controller to satisfy the target source data rate by modulating the video-processing parameters (i.e., the threshold and the QF). Using proportional (KP),

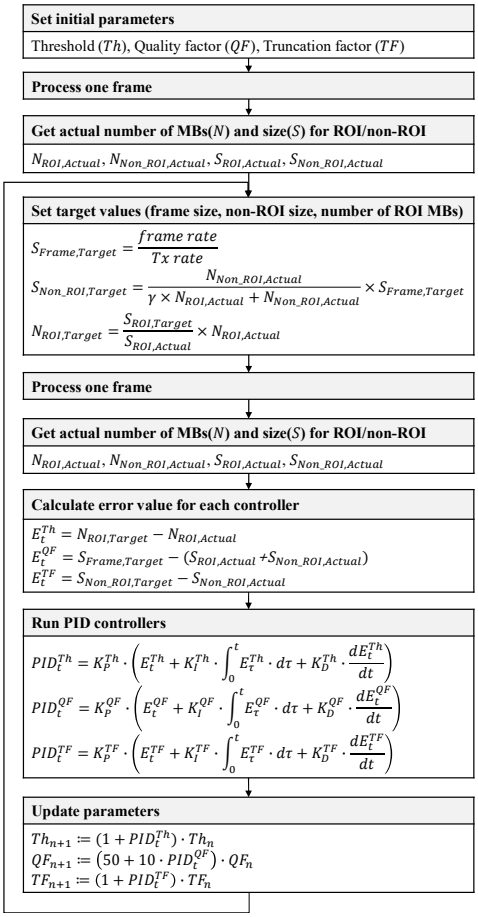


Figure 5.7: Flowchart of the proposed on-line rate controller.

integral (KI), and differential (KD) coefficients, a PID controller modifies a control parameter based on the error value, which is the difference between the target and the previous value of a variable. As our system has two control parameters (i.e., the threshold and the QF) and one variable (i.e., the source data rate), Using three PID controllers, our rate controller modulates the non-ROI coding parameter (the truncation factor) as well as the ROI coding parameters [Figure 5.6(b)]. The first PID controller modifies the threshold so that it can maintain the target number of encoded MBs. The QF is controlled by the second PID controller to maintain the frame size, and the truncation factor is controlled by the third PID controller to maintain the non-ROI data size per frame.

The procedure of the proposed rate controller is illustrated in Figure 5.7. First, the

number of MBs and size for the ROI/non-ROI are obtained from the first frame processing with the initial parameters. Based on these values, target values for the next frame are set. To set the target data size of the non-ROI, I introduce a weight factor (), which indicates how many bits to allocate for one ROI MB compared to one non-ROI MB. After the ratio of the numbers of ROI/non-ROI MBs are obtained, it is multiplied by the weight factor to determine the ratio of the target ROI/non-ROI data size. Therefore, the weight factor can also be used as a knob to control the quality/size difference of the ROI and non-ROI. After the target values are configured, one frame is processed, and actual values for the number of MBs and size are obtained. Then the error values (difference of target and actual values) are fed into the PID controllers, which update the parameters for the next frame. Since the controller determines the threshold for the current frame based on the error value of the previous frame, the target number of MBs may not be satisfied if the ROI content varies from frame to frame. However, the variation in frame size can be reduced by controlling the QF using the second PID controller.

The first PID controller assumes the QF=50 because the second controller will control the QF in a range of 1-100 depending on the actual number of MBs determined by the first controller. The error value for the threshold control, E_t^{Th}, defined as the difference between the target and actual number of MBs in the n-th frame at time t, is supplied to the PID controller

$$PID_t^{Th} = K_P^{Th} \cdot (E_t^{Th} + K_I^{Th} \cdot \int_0^t E^{Th} \cdot d + K_D^{Th} \cdot (dE_t^{Th})/dt) \tag{5.2}$$

where K_P^{Th}, K_I^{Th}, and K_D^{Th} are the proportional, integral, and derivative coefficients of the first PID controller, respectively. The threshold value for the next frame, $Th_(n+1)$, is adjusted by

$$Th_{(n+1)} := (1 + PID_t^{Th}) \cdot Th_n \tag{5.3}$$

Since the controller determines the threshold for the current frame based on the error value

of the previous frame (i.e., the deviation between the target number of MBs to encode and the actual number of ROI MBs in the previous frame), it may not satisfy the target number of MBs if the distribution of the activity level changes from frame to frame, leading to variation in the source rate. However, this variation can be reduced by controlling the QF using the PID controller in the second stage. For example, if the first PID controller determines the threshold so that the actual number of MBs exceeds the target number of MBs, the actual data size will be larger than the target data size. Then, the QF will be decreased to satisfy the target data size by the second PID controller, which is given by

$$PID_t^{QF} = K_P^{QF} \cdot (E_t^{QF} + K_I^{QF} \cdot \int_0^t E^{QF} \cdot d + K_D^{QF} \cdot (dE_t^{QF})/dt) \tag{5.4}$$

where E_t^{QF} is the error value for the QF control, defined as the deviation between the target and actual source data rate. The error value adjusts the QF for the next frame, $QF_{(n+1)}$, by

$$QF_{(n+1)} := (50 + 10 \cdot PID_t^{QF}) \cdot QF_n. \tag{5.5}$$

To design a PID controller with desirable control performance, we need to properly tune the three coefficients, K_P, K_I, and K_D. In this work, I determine the coefficients through experimental analysis, i.e., by varying the coefficients of the two PID controllers between 0.02 and 4. Figure 5.8(a) shows that, as we increase KP, the required buffer size decreases since a large KP leads to a large change in the parameters for a given change in the error. However, as KP increases, the quality of the ROI degrades because more ROI MBs are dropped or encoded in a lower quality. Moreover, the system with too large KP may become unstable, generating higher peak-to-peak variation in the source data rate. The integral coefficient, KI, accelerates the movement of the process towards the target, but too large KI may cause the value to overshoot the target. Thus, as Figure 5.8(b) shows, a controller with too small KI modifies the parameters too slowly, while the overshoots created by too large KI increases buffer size. Although another coefficient, KD, can improve settling time and

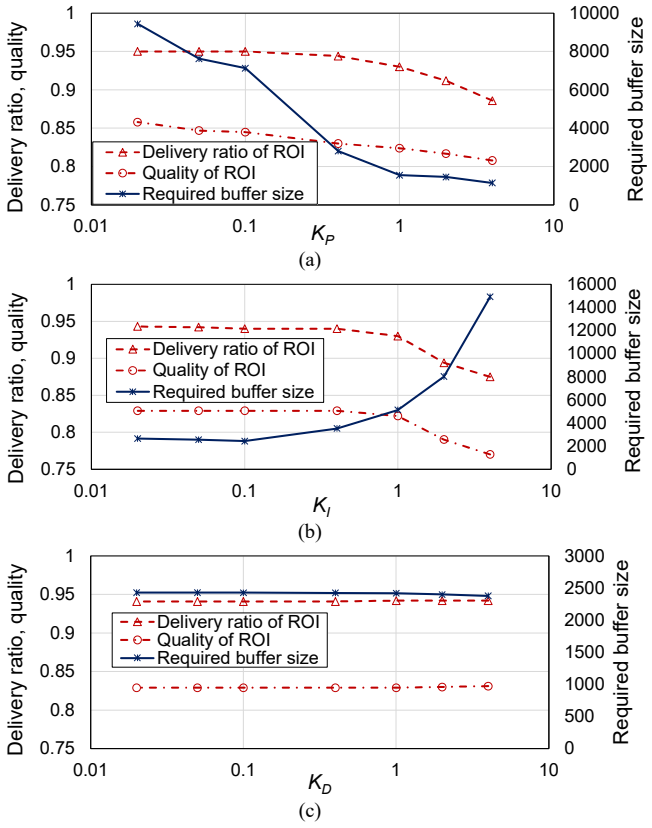


Figure 5.8: Simulation results of the PID-based controller with varying (a) KP, (b) KI, and (c) KD.

stability of the system, it makes a minor change in the performance of the controller [Figure 5.8(c)]. Based on the experiments, I set the values of (K_P, K_I, K_D) in two controllers to (1, 0.1, 0.1). These values work for all four video sequences at the range of target source rates between 160 and 4,500 bytes/frame.

5.2.4 Experimental Results

Comparison with a Fixed-Parameter System

Figure 5.9 shows the source data rates (data size per frame) controlled to satisfy a transmission rate of 400 bytes/frame and corresponding buffered data size for four video sequences. It can be observed that data buffering occurs when the source data rate exceeds the target

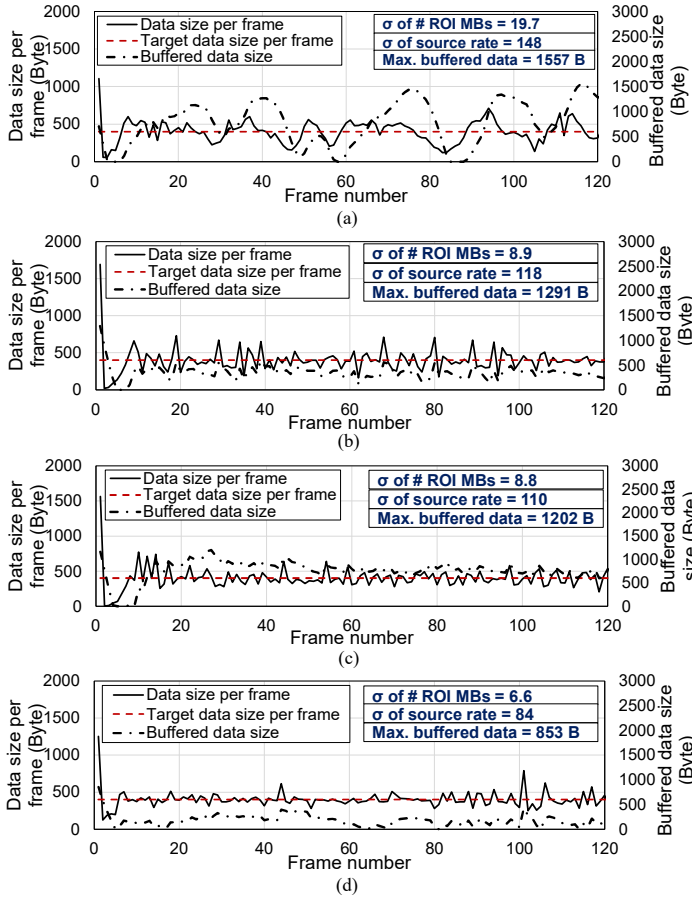


Figure 5.9: Simulation results of the PID-based controller with (a) traffic video, and (b) atrium video, and (c) hall monitor video, and (d) bridge video.

rate. Also, a video stream with more variation in the source data rate requires a larger buffer (larger maximum amount of buffered data). For ex-ample, the traffic video, which shows more fluctuation in the source data rate (standard deviation=148), requires a larger buffer (1,557 bytes) [Figure 5.9(a)], while the bridge video, which has less variation (standard deviation=84), requires a smaller buffer (853 bytes) [Figure 5.9(d)]. The reason why video sequences differ in the source rate variation is that they have different frame-by-frame variations in the number of ROI MBs. A video stream with more variation in ROI MBs leads to a larger variation in the source rate because of the nature of a PID controller. That is, if the current frame has more ROI MBs than the previous frame, we should increase the

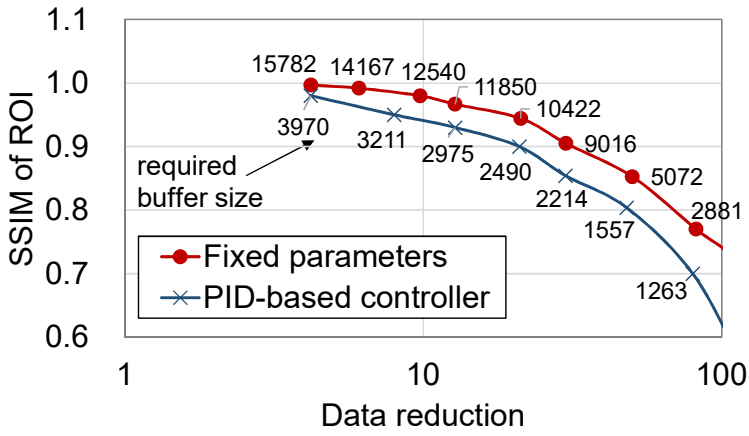


Figure 5.10: Rate-quality characteristics of the fixed-parameter scheme and the PID-based controller

threshold or decrease the QF to satisfy the target rate. However, since a PID controller determines the parameters based on previous observations, varying number of ROI MBs causes deviation in the source rate until the controller settles to the parameters that satisfy the target rate. As Figure 5.9 shows, the video with the largest variation in the number of ROI MBs (traffic video) shows the largest fluctuation in the source rate.

I examine the ROI quality and the required buffer size by varying the target source data rate using traffic video. Figure 5.10 shows that the PID-based controller requires 74-83% smaller buffer than the fixed-parameter scheme, with a loss of the ROI quality less than 0.07. For the same target quality (say, 0.95), the PID controller requires 69% less buffer than the fixed parameter case.

Since the required buffer size is a function of the input video characteristics [Fig5.9] and the target source rate [Figure 5.10], it is difficult to be estimated a priori. Moreover, a large buffer adds significant area/energy overhead to a sensor node. Therefore, a more practical approach for resource-constrained sensor node is to have a small fixed buffer or no buffer. If I assume a system without a buffer, where the source data rate higher than the channel data rate leads to random packet drop, the PID-based controller enhances the ROI quality for all four video sequences [Figure 5.11]. This is because, when the source rate exceeds the target rate for a given frame, the PID controller tries to minimize random drop

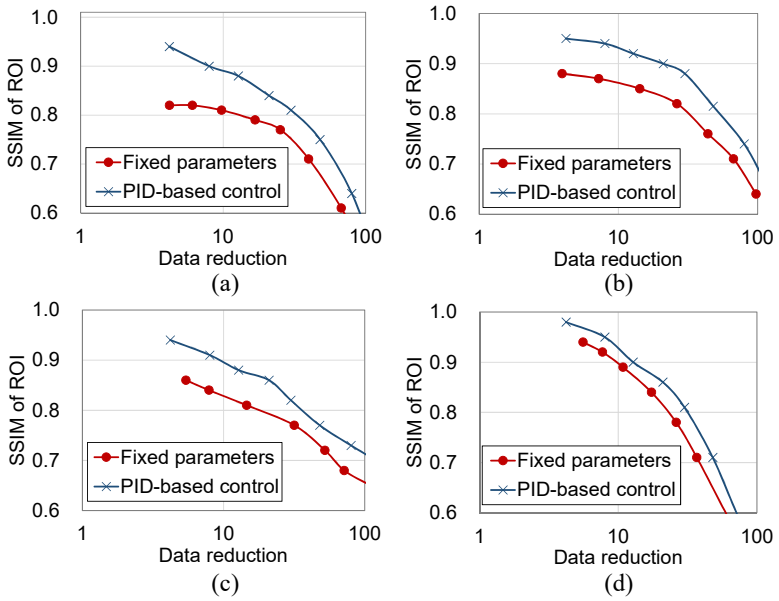


Figure 5.11: Rate-quality characteristics of the fixed-parameter scheme and the PID-based controller assuming a system without a buffer with (a) traffic video, (b) atrium video, (c) hall monitor video, and (d) bridge video.

by using a higher threshold and/or a lower QF than the fixed-parameter scheme. Using the rate-quality curves assuming no buffer, I study the ROI quality under varying channel conditions.

As Figure 5.12 shows, if we assume a system with an infinite buffer (as in the case of Figure 5.4), the PID-based controller leads to a lower ROI quality than the fixed-parameter scheme (the condition assumed in Figure 5.4). However, when a system with no buffer is assumed, the PID-based controller provides higher ROI quality by minimizing random drop under the varying channel conditions.

Comparison with Existing Rate Controller Schemes

I compare the performance of the proposed rate controller with the H.264/AVC intra (all I-frames) and inter (group of pictures (GOP) structure: IBBPBBPBBPBBP) modes at a target transmission rate of 100 Kbps [Figure 5.13]. In both modes, H.264/AVC controls the QP to minimize quality degradation while satisfying the average data rate. Figure 10 shows that

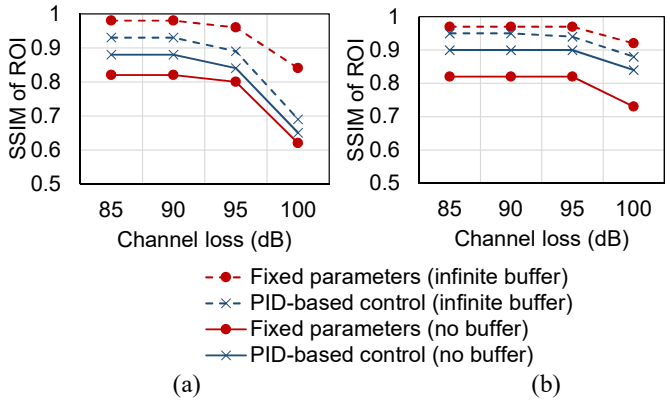


Figure 5.12: Quality of ROI under varying channel condition with (a) traffic video and (b) atrium video.

the proposed controller provides comparable rate-control performance as the H.264 intra mode by adaptively controlling three parameters. However, the inter mode shows large fluctuation in the encoding rate mainly due to the inherent nature of H.264/AVC encoding where I-frames are encoded independently while B- and P-frames are compressed more through motion estimation. This variation in frame size, however, is not resolved by the rate controller, which first determines a target bit rate for a GOP and then performs detailed bit allocation to pictures and MBs. Therefore, in many cases, a large data amount is allocated for earlier I-frames, while the bit rates of the remaining frames of the GOP are decreased dramatically, resulting in large fluctuation in encoding rate [108]. Instead of satisfying the average rate for several frames, the proposed controller tries to satisfy the target rate for each frame for less fluctuation of the encoding rate.

Figure 5.14(a) shows that, at a given target rate, H.264/AVC provides higher quality than the proposed controller. However, due to the significant computation overhead, H.264/AVC consumes more system energy than the proposed approach, as illustrated in Figure 5.14(b). Even if the ROI-based coding method is added to H.264/AVC for higher ROI quality, it will further increase energy consumption. Although the proposed approach requires more data transmission to achieve the same quality, with its significantly low com-

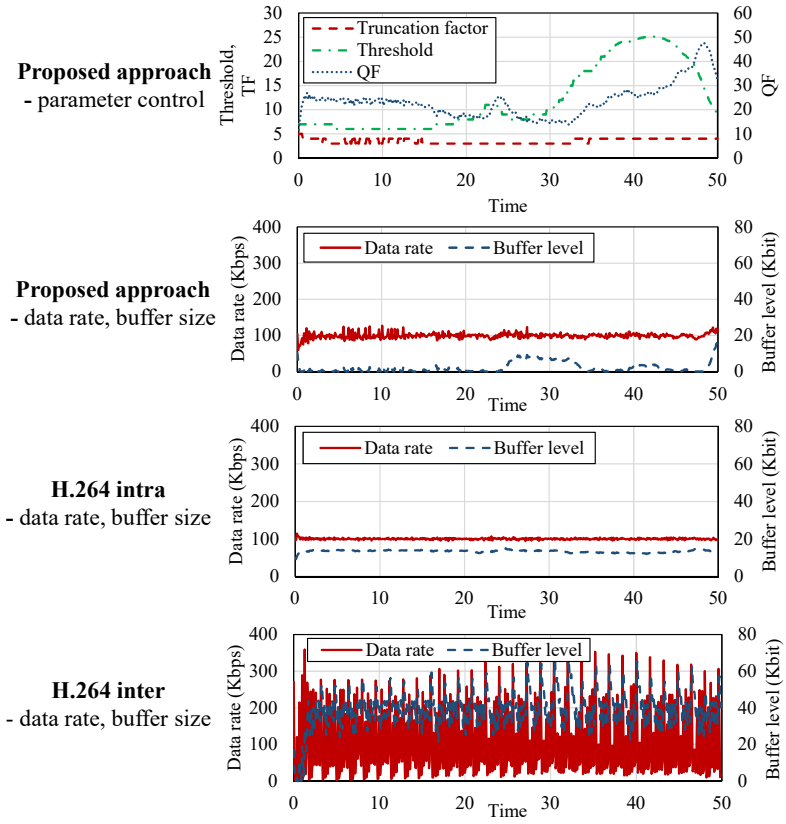


Figure 5.13: Rate-control performance of each approach.

putation energy and reduced buffer overhead, it consumes less system energy than other approaches at the target ROI quality of 0.8 [Figure 5.14(c)]. Note that the system analysis presented in this work considers relatively power-hungry 2.4GHz RF transmission module. The recent advancements in radios for IoT sensor nodes have focused on enhancing transmission power efficiency [78], creating a stronger need to reduce computation energy in such sensor nodes. Therefore, the proposed approach is suitable solution for emerging IoT sensor platforms with low-power radio.

FPGA Demonstration

I synthesize the proposed system into a Xilinx Virtex-5 FPGA. Figure 5.15(a) shows resource utilization and computation energy of the FPGA implementation. Figure 12(b) shows the system demonstration framework, where I validate the functionality of the pro-

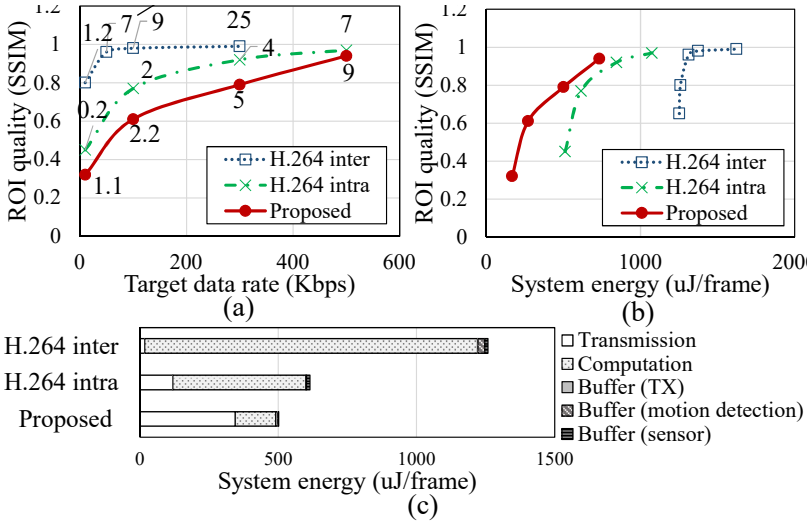
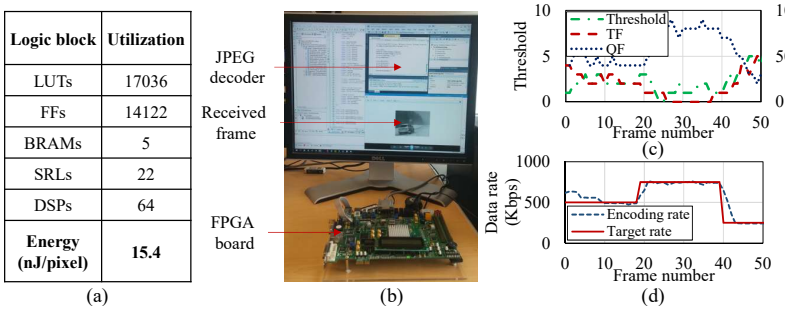


Figure 5.14: ROI quality vs. (a) target data rate and buffer size, and (b) system energy. (c) System energy breakdown at ROI =0.8.



Logic block	Utilization
LUTs	17036
FFs	14122
BRAMs	5
SRLs	22
DSPs	64
Energy (nJ/pixel)	**15.4**

Figure 5.15: (a) Resource utilization and (b) test environment of the FPGA demonstration. (c) Controlled parameters and (d) data rate.

posed design that controls parameters [Figure 5.15(c)] to satisfy varying target rate [Figure 5.15(d)].

5.2.5 Effect on the Neural Network Inference Accuracy

To compare the energy-accuracy scalability of the various rate controller designs, I used the same test points used for the AlexNet classification. Figure 5.16 shows that the proposed ROI-based rate controller shows better performance under the given energy budget with the classification system as well.

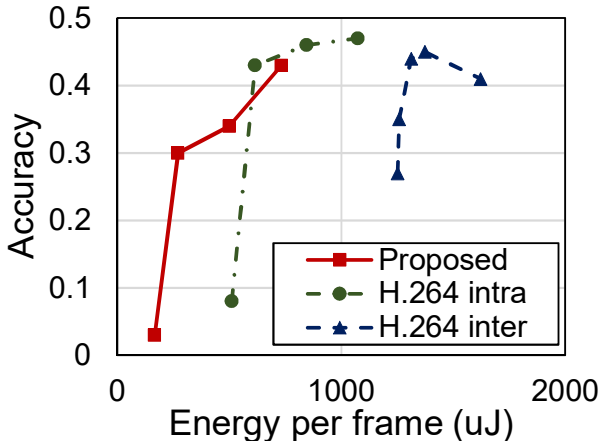


Figure 5.16: Comparison of energy-accuracy scalability for different data rate control algorithms.

5.2.6 Summary of the Section

This section presented an energy-efficient source data rate controller for ROI-based coding scheme to enhance the ROI quality with lower system energy consumption. The on-line rate controller modulates the parameters of the ROI-based coding scheme to match the encoded data rate and transmission data rate under the variations in channel bandwidth and input video content.

CHAPTER 6

RESOURCE-EFFICIENT DEEP NEURAL NETWORKS

6.1 Adaptive Weight Compression for Memory-Efficient DNNs

6.1.1 Introduction

Artificial neural networks (ANNs) are being successfully applied to many applications including computer vision [91], speech recognition [109], and financial prediction [110], etc. There is a growing interest in deploying complex ANNs at the edge devices such as cameras, smart phones, sensor nodes for intelligent data processing [5][6][7]. The energy and resource constraints in such edge devices create intriguing challenges to deploying complex ANNs. Significant past efforts have explored techniques to reduce computation energy of ANNs, for example, using lower bit precision [111], computation re-use [49], and approximation techniques [56].

An ANN contains a large number of synaptic weights, and the size of the weights increases with the network connectivity and the number of layers. In particular, fully connected networks require a large number of weights to compute each neuron state, resulting in a low value of the number of operations per byte of data (ops/byte). Figure 6.1 shows the memory demand for different ANNs including a multilayer perceptron (MLP) for the MNIST dataset [112] and three different convolutional neural networks (CNNs) including LeNet-5 [113] for MNIST dataset, and AlexNet [91] and ResNet-152 [99] for ImageNet dataset. The figure shows that the memory demand increases with increasing complexity of the network. Likewise, for a given network, the memory demand increases with the increasing input space as illustrated in Figure 1(b).

The memory demand is a key challenge for application of ANNs, especially for resource-constrained platforms such as mobile systems [52]. It is evident that even for a moderate

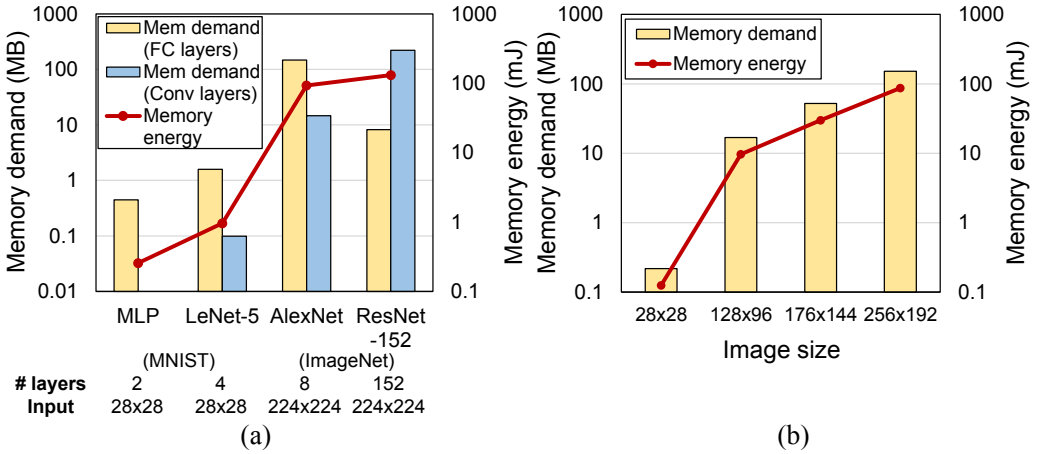


Figure 6.1: Memory requirement and energy for different neural networks. (a) MLP and CNNs for MNIST and ImageNet datasets, (b) MLP with different input image size. Memory energy includes only access energy from a DDR3 off-chip memory.

image size and simple networks, an on-chip cache of a general mobile processor is not sufficient to store all the weights, so the weights should be stored in an external memory. The demand for larger external memory will increase the system cost. Moreover, the system performance and energy will strongly depend on the time/energy of data movement between off-chip memory (DRAM and/or SSD) and a computation unit. The energy for accessing data from external memory (70 pJ/bit for DDR3 [114] and 3.6 nJ/bit for SD card [115]) is far larger than that of on-chip memory access (0.16 pJ/bit for 28nm SRAM [116]) or computation (16 pJ/16-bit multiply-accumulate operation with 28nm process [117]). Consequently, an ANN inference engine that stores and accesses a large number of weights in off-chip memory incurs appreciable energy cost for data movement, resulting in degradation of the energy-efficiency (Figure 6.1). The processing-in-memory architecture can partly address the challenge of data movement cost, but it requires new memory and packaging technologies [116][118][119]. The orthogonal and complementary approach is to reduce the storage requirement by compressing the weights while preserving the accuracy of the network (see section II for details). However, most of these approaches require modification of a training process [53] or retraining (fine-tuning) [52] to minimize the ac-

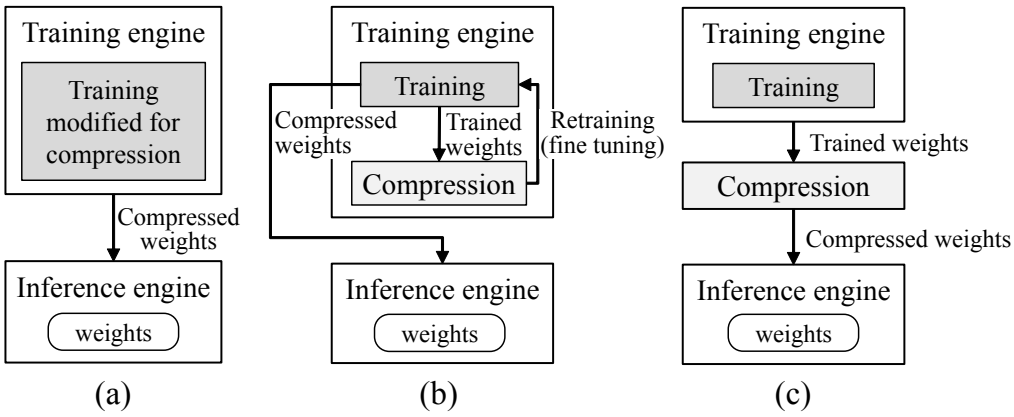


Figure 6.2: Three different weight compression schemes. (a) Compression requiring modified training algorithm, (b) Compression requiring fine tuning, and (c) proposed training-independent compression scheme

curacy loss [Figure 6.2(a), (b)]. When we want to apply uncompressed, pre-trained ANNs to inference engines, the compression methods requiring re-training may create additional challenges.

This section presents design of an inference engine based on compression of the already-trained weights without modifying the baseline training process and/or using re-training [Figure 6.2(c)]. As Figure 6.1(a) shows, while memory demand of earlier CNNs such as LeNet-5 and AlexNet is dominated by fully connected layers, recent neural networks such as ResNet have very deep convolutional layers that dominate the memory demand. Therefore, this work presents the compression method that can be applied to both convolutional and fully-connected layers. Inspired by smoothness and spatial locality of the weights, I propose to apply an image compression technique on the weight matrix (instead of applying data encoding techniques such as pruning on the individual weight). The key contributions of this section are:

- I propose to apply a JPEG image encoding algorithm to compress the weight matrix as a single image. I present the compression techniques for both the convolutional layers and the fully connected layers.

- For higher compression with minimum loss of accuracy, I adaptively control the

JPEG encoding parameter (quality factor, QF) depending on the error-sensitivity (gradient) or the entropy of the weights.

- I analyze the effect of various algorithmic parameters such as 1-D/2-D JPEG processing, number of QF values, and the block formation method on the compression performance and overhead. I also analyze how bit truncation techniques combined with the proposed method affect the compression performance.

- I present the design of an inference engine with an embedded JPEG decoder. I study the impact of weight compression on the memory requirement, throughput, and energy consumption of the inference, by considering memory access energy/latency and decoder overhead.

The compression performance of the proposed approach is demonstrated with MLP and CNNs for several benchmark datasets. At 1% degradation in recognition accuracy, the proposed method achieves 63.4X compression for MLP and 31.3X for LeNet-5 with the MNIST dataset, and 15.3X for AlexNet and 10.2X for ResNet-50 with ImageNet. For system-level performance and energy analysis, a digital neural network inference engine with weight compression/decompression modules is synthesized in 28nm CMOS technology. The proposed approach shows 3X higher effective memory bandwidth and 22X lower system energy for inference of MLP with MNIST.

6.1.2 Adaptive Image Compression of Weights

JPEG-based Image Encoding

JPEG [120] is a common encoding method for compressing images. A source image into JPEG is first divided into 8x8 block, and each block is transformed into the frequency domain by discrete cosine transform (DCT). Then, the 64 DCT coefficients are quantized in conjunction with a quantization table, which is specified by the quality factor (QF) in a

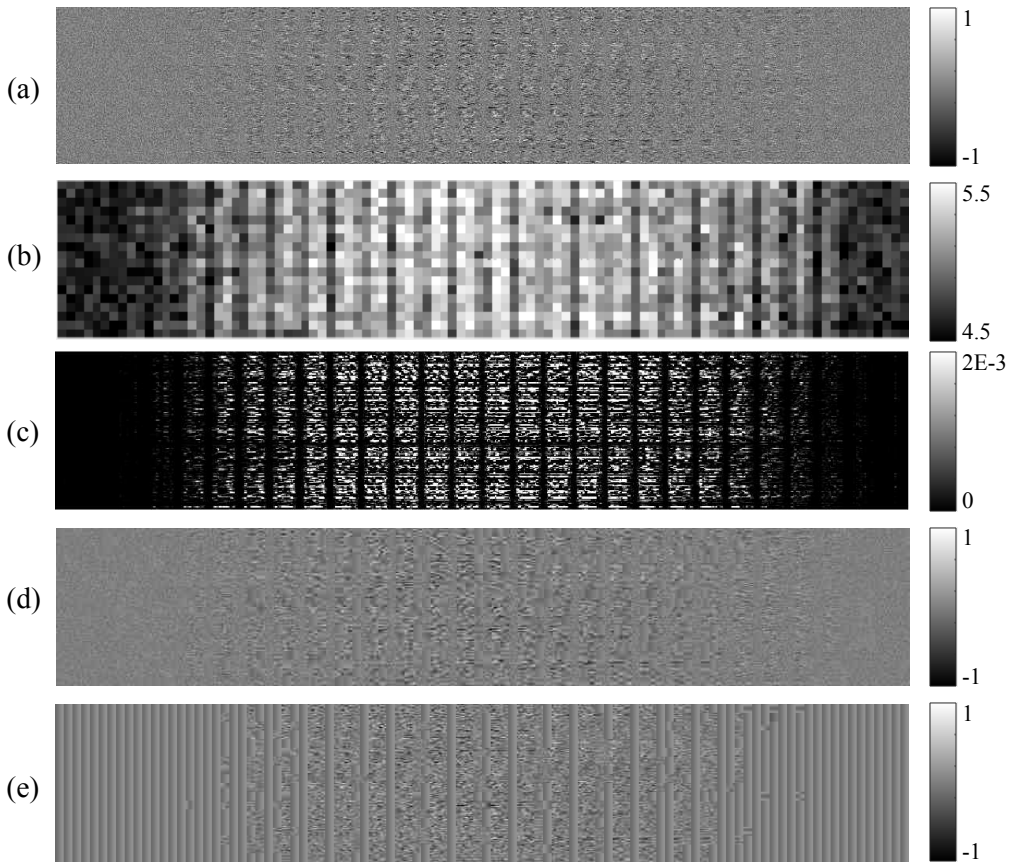


Figure 6.3: Visualization of a weight matrix of MLP for MNIST. (a) Original weight matrix, (b) average entropy, (c) absolute error sensitivity, (d) compressed matrix using JPEG with single QF, and (e) compressed matrix using JPEG with multi QF.

range of 1-100. The quantization process is a principal source of losing the original information, and the QF is used to control the lossiness. Higher QF preserves more information with less compression, while lower QF lose more information with higher compression ratio. The final step, the entropy coding, achieves additional compression by losslessly encoding the quantized DCT coefficients based on their statistical characteristics. The standard JPEG encoder compresses 2-D input blocks using 2-D DCT and quantization. This section proposes modified 1-D JPEG encoding for better weight compression performance, which will be discussed later.

Motivation

The weight parameters in each layer of neural networks can form a 2-D matrix, and can be represented as an image with each pixel being the corresponding weight value. Let us take an example of an MLP network trained for MNIST dataset, which has a 784-neuron input layer, a 144-neuron hidden layer, and a 10-neuron output layer. Figure 6.3(a) shows visualization of a 784 x 144 matrix of weights in its first fully-connected layer. As seen in the figure, a large portion of the weight image is smooth, and has certain types of pattern (redundancy). This is known to happen due to the nature of spatial locality (local pixel correlation) in the input dataset of the network [51]. As images with smoothness can be efficiently compressed by image encoding techniques, I apply JPEG encoding for compressing the weight parameters. The compression performance achieved by image encoding is related with the entropy theory [121]. The entropy is the statistical measure of the amount of information contained in an image, and used to describe its 'business' or randomness. Areas with smoothness or redundancy generally have lower the entropy and can be compressed more. Figure 6.3(b) shows the entropy distribution of the sample weight matrix in a unit of an 8x8 block. It can be observed that its entropy (required bits per weight) distribution spans from 4.5 to 5.5. Since the original weights are represented with 32 bits per weight, the entropy distribution indicates rooms for compression. When I plot the error sensitivity (gradient) of the weights together [Figure 6.3(c)], it can be seen that the areas with higher error sensitivity tend to have higher entropy. Consequently, the image encoding will inherently assign more bits to areas that are relatively more critical for accuracy. Although image encoding techniques can help preserve the critical information, loss of information is inevitable because of the quantization process (DCT in JPEG). The weights encoded and decoded by the JPEG algorithm with a single QF value of 20 [Figure 6.3(d)] show that high-gradient areas are also distorted. To minimize the distortion of the important areas, we can use an adaptive image compression technique commonly used for region-of-interest image coding [22]. I propose to use a higher QF value for accuracy-

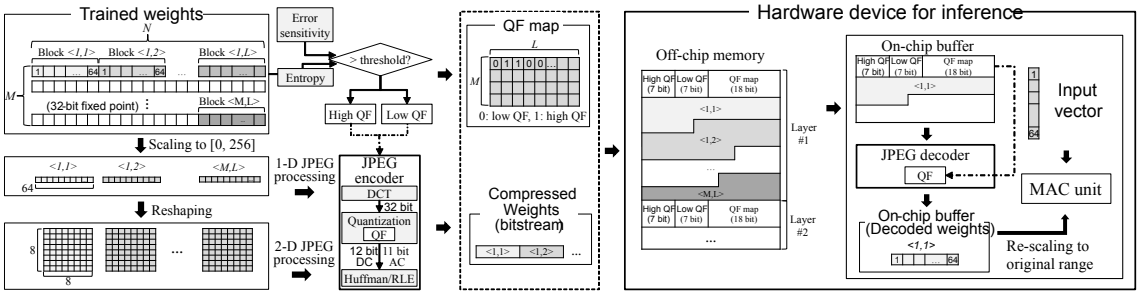


Figure 6.4: Overall process of the proposed compression approach. While this figure illustrates both 1-D and 2-D JPEG processing, I use 1-D JPEG processing as a baseline approach.

critical regions, and a lower QF value for less critical regions. By differentiating the quality factor, blocks with higher sensitivity can be quantized less for higher accuracy, while those with lower sensitivity can be compressed more for higher compression [Figure 6.3(e)].

Adaptive Weight Compression Technique

Overall encoding and decoding processes are described in Figure 6.4. First, the 32-bit fixed point weight values are scaled into [0, 256] to make them grayscale pixel values for a JPEG encoder. The weight values are first shifted into the range of [-128, 128] by moving the decimal point towards left. The number of bits that the decimal point is moved to the left is defined by the scale factor, which is computed as

$$n = log_2 \left(128 / \max \left(abs \left(w_{ij}\right)\right)\right). \tag{6.1}$$

After setting the range of the values to be [-128, 128], 128 is added to make them in the range of [0, 256]. Even after scaling, the bit width is kept 32 bits as I only move the decimal point. After scaling, every 64 elements of each column of the weight matrix are used to form a JPEG input block. I encode the weights in a column-wise manner to minimize the required on-chip cache size, as will be discussed in the next section. For the standard 2-D JPEG encoding, the 64-element vectors are reshaped into 8x8 sub-blocks. For better compression performance, I present the modified 1-D JPEG processing approach,

where the vectors can be directly supplied without reshaping. Inside the JPEG encoder, the weight blocks are first processed by discrete cosine transform (DCT). The 32-bit DCT coefficients are then quantized by the JPEG quantization table and rounded to integer to make a 12-bit DC element and 11-bit AC elements. The quantized values are processed by the run-length encoder and the Huffman encoder to generate compressed bit-stream, which is stored in the memory of an inference engine. By default, I do not reduce the bit precision of the weights before compression. Bit precision control can be inserted at the various stages of the process; the weights can be trained with low precision, or trained weights can be truncated before encoding/after decoding. The effect of bit truncation combined with JPEG compression is analyzed in Section IV.E. After the compressed weights are decoded at the inference engine, they are in the range of [0, 256], so they need to be scaled back to the original range. The output values from the JPEG decoder are subtracted by 128, and the decimal point is moved by the scale factor, which was obtained at the encoding stage. For adaptive compression, we need the information on the accuracy impact of each corresponding block, which can be determined by the average error sensitivity or the entropy. Based on this information I adaptively allocate the QF of the block. First, the threshold is chosen to be the error sensitivity or the entropy value that equally divides the weight blocks into two groups. If the average error sensitivity or the entropy is higher than the predefined threshold, high QF value is chosen for quantization, while low QF value is used for blocks with the accuracy impact value lower than the threshold. Therefore, the high/low QF values can be used as knobs to achieve the desired compression ratio and accuracy level. To decode the weights that are adaptively compressed, we need to store a QF map that represents the QF selection of each block and the QF values. I discuss the effect of the number of QF values on the performance and overhead later.

Obtaining Accuracy Impact of the Weights For adaptive QF allocation, I consider the error sensitivity and the entropy [Figure 6.5]. The error sensitivity can be obtained through

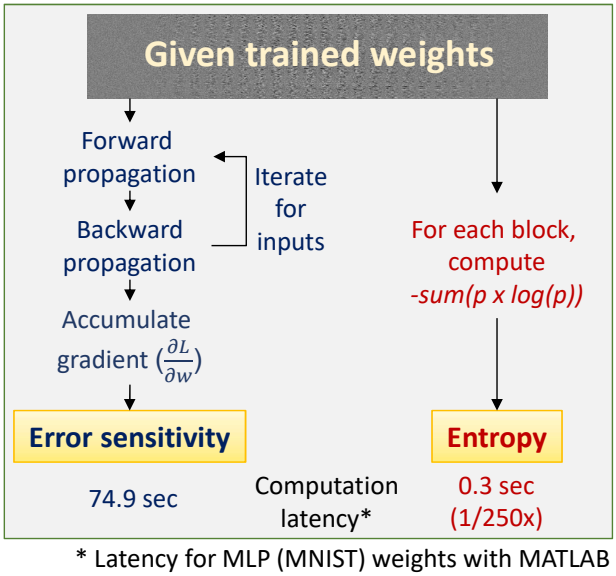


Figure 6.5: Computation process of the error sensitivity and the entropy

the back-propagation process during training. In the supervised learning of feedforward neural networks, each synaptic weight w_{ji} is updated so that the error in the output layer can be minimized [122]. To update the weight, the backpropagation algorithm computes the sensitivity (gradient) of each weight to the output error ($\partial \mathrm{E}/\partial w_{ji}$). Using the error sensitivity, each weight is updated as follows:

$$w_{ji}\,(t+1) = \; w_{ji}\,(t) - \eta \frac{\partial E\,(t)}{\partial w_{ji}\,(t)} \tag{6.2}$$

where η is learning rate E(t) is the error function, $w_{ji}(t)$ is the current weights and $\partial E\,(t)\,/\partial w_{ji}\,(t)$ is the gradient at iteration t, and $w_{ji}(t+1)$ is the updated weights at iteration t+1. The gradient for each weight can be computed by the chain rule using back-propagation. In recent studies, the gradient information has been utilized to design low-power neural networks through approximate computing [117]. In this section, I propose to use the error sensitivity to determine the compression level of each weight block. The entropy is the statistical measure of the amount of information (randomness) contained in an

image, and can be used to characterize the texture of the image. Entropy can be computed as

$$E = -\sum_{i} p_i \log_2 p_i \tag{6.3}$$

where p_i is the probability associated with gray level i. In practical, p_i can be computed by counting the number of pixels per each histogram bin. When compression is performed in conjunction with training, the error sensitivity will be readily available for QF allocation. However, when the pre-trained network is given for compression, the forward/backward propagation should be performed again to obtain the error sensitivity. The primary advantage of using the entropy is that the entropy can be easily obtained from trained weights with much less computation than the error sensitivity. Our simulation with MLP using MATLAB environment shows that computation of the entropy takes 150X less time than computing the error sensitivity. This is because the error sensitivity is computed through complex backpropagation algorithm while entropy computation involves simple histogram-based computation.

Block Formation for Fully-Connected Layer Compression The block formation method for the JPEG encoder should consider efficient data movement in an inference engine. The inference engine first decodes the compressed weights fetched from the off-chip memory, to perform the multiply-accumulate (MAC) operations with input neurons. To compute each output neuron, a column of the weights should be multiplied with the corresponding input column vector. The amount of decoded weights loaded into an on-chip cache depends on how the weights are formed into JPEG blocks.

The most straightforward approach is a block-wise formation, in which the weight matrix is divided into 8x8 blocks and encoded, as shown in Figure 6.6(a). To obtain one row of weights in this approach, eight columns (i.e. one column of 8x8 blocks) should be decoded. Assuming an inference engine with a single MAC unit, this approach requires on-chip memory size of 8 columns of weights to store the decoded weights. To reduce the

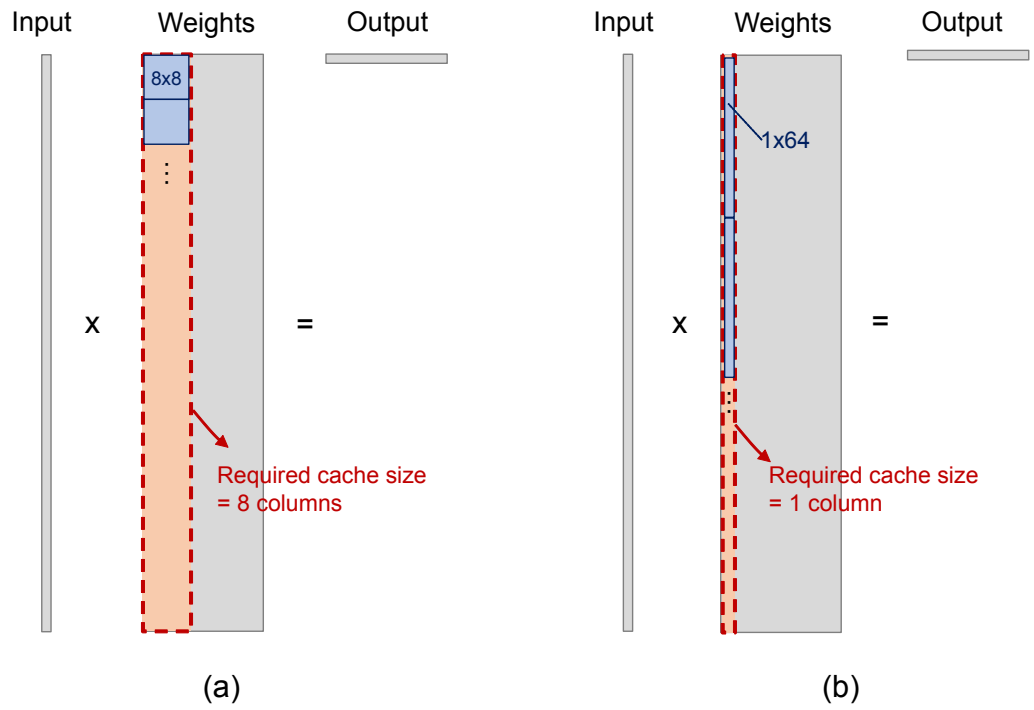


Figure 6.6: Block formation approaches. (a) Block-wise approach, (b) column-wise approach.

required on-chip memory size, I perform column-wise block formation as shown in Figure 6.6(b). The column-wise scheme takes 64-element vector of the weights and reshapes it into 8x8 blocks for encoding. Then each column can be separately decoded and computed with the input vector, requiring only one-column size on-chip cache. In the column-wise encoding approach, we can reduce the on-chip memory overhead per MAC by having multiple MAC units in the inference engine. For example, eight MAC units can be used to process each of eight columns, requiring one-column size cache per one MAC unit. However, having multiple MAC units also reduces the on-chip memory overhead per MAC unit of the column-wise encoding because MAC units can process eight patches of one row in parallel. Therefore, the column-wise encoding scheme requires 8X smaller on-chip memory than the block-wise encoding scheme for configurations with any number of MAC units.

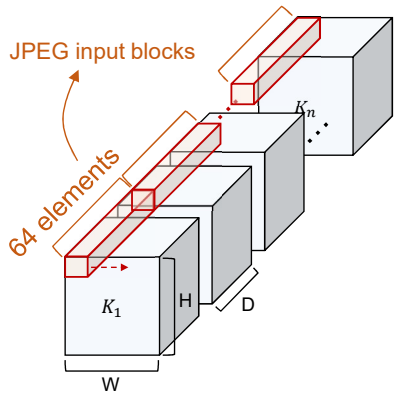


Figure 6.7: JPEG input block formation from n convolutional layer kernels with width W, height H, and depth D.

Block Formation for Convolutional Layer Compression Most of the recent CNNs utilize 3x3 or 5x5 kernels in their convolutional layers. Therefore, the number of weight elements in one channel of the kernel is generally smaller than the JPEG input block size (64 elements). As a single channel of kernels cannot form a JPEG input block, we need to re-organize the weight blocks from the kernel matrix. As illustrated in Figure 6.7, I take 64-element input vectors along the channel and depth direction of the kernel weights. By using this approach, the elements in the same location of the kernel will be mostly placed in the same JPEG input block. Therefore, this approach ensures that the elements in a block have more similarity, which will result in higher compression performance.

1-D JPEG processing Since the JPEG input blocks are casted from 1-D weight vector, the standard 2-D JPEG processing may not properly capture the redundancy along the weight vector. Therefore, I propose using modified 1-D JPEG encoding. Figure 8(a) shows the original 2-D JPEG compression flow with a 2-D input matrix, and Figure 6.8(b) shows the modified compression flow that processes 1-D input vector. Instead of reshaping the 64-element weight vector into 2-D 8x8 matrix, the modified procedure takes the original 1-D vector and performs 1-D DCT. The DCT coefficient vector is then quantized by the

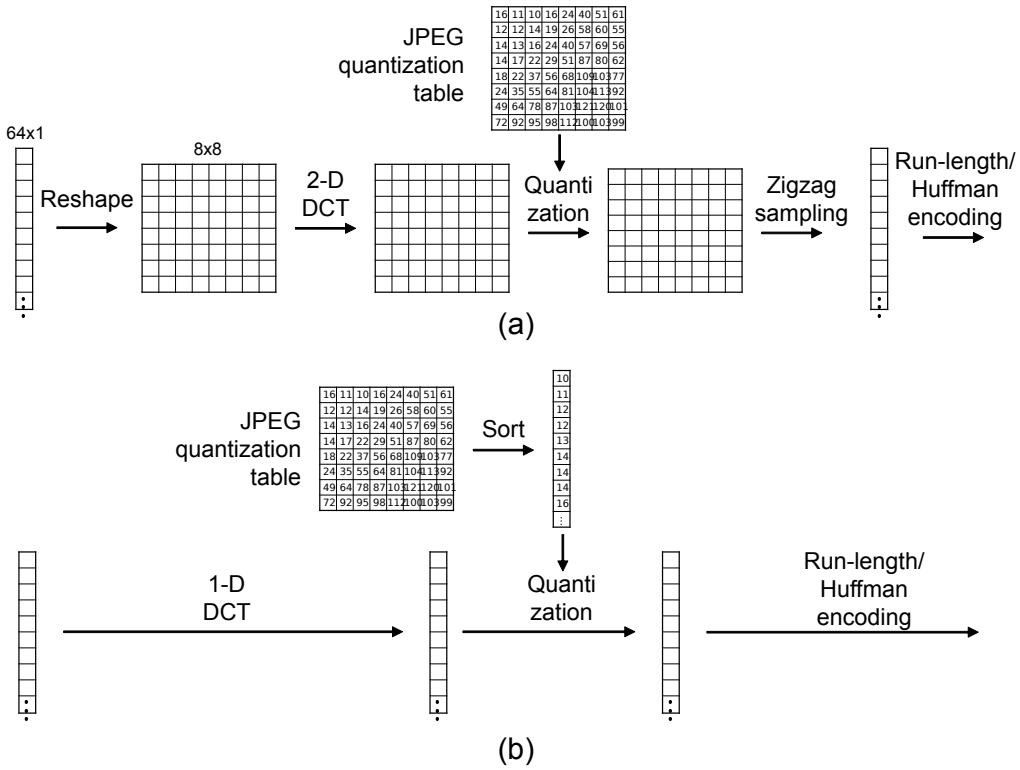


Figure 6.8: JPEG encoding process. (a) The standard 2-D JPEG encoding process and (b) the modified 1-D JPEG encoding process.

modified 1-D quantization table. The 1-D quantization table is generated by arranging the elements of the original JPEG quantization table in an increasing order. After quantization, the quantized element vector is directly processed by the run-length encoding and Huffman encoding without the zig-zag sampling. The 1-D JPEG encoder is expected to provide better compression performance in compressing 1-D weight vectors. Moreover, the 1-D JPEG decoder requires less computation than the 2-D JPEG decoder. For 2-D IDCT computation, 1-D IDCT is applied on eight of 8-element vectors, once along the vertical direction and another once along the horizontal direction [123]. Therefore, 2-D IDCT processes 128 elements, while 1-D IDCT can be computed by processing 64-element vector once. Also, it does not require reshaping and zig-zag sampling.

6.1.3 Analysis of Algorithmic Performance

Experimental Framework

To validate the proposed method, I use various networks and datasets. First, I consider a typical MLP-based neural network with three benchmark datasets: MNIST [112], CNAE-9 [124], and SVHN [38]. The network is implemented in MATLAB, and it has three layers for MNIST (784-144-10), four layers for CNAE-9 (856-144-64-9) and SVHN (1024-144-64-10). I also consider CNNs including LeNet-5 for MNIST, AlexNet and ResNet-50 for ImageNet dataset. For these networks, I use the Caffe framework and reference trained network models.

Performance Analysis with Algorithmic Parameters

The proposed approach involves algorithmic parameters including the block formation method, the number of QF values, 1-D/2-D JPEG processing, and the QF selection approach. In this section, I analyze the effect of each parameter on compression performance and efficiency. Based on the analysis, I determine the default parameters for the rest of the simulation.

Number of QF values To analyze the effect of the number of QF values on compression performance, I plot the average gradient and compression ratio of the blocks in the weight matrix of MLP for MNIST. When I use JPEG compression with a single QF value (QF=20), blocks with lower error sensitivity (lower gradient) tend to be compressed more [Figure 6.9(a)]. By applying JPEG encoding with two QF values (high QF=25, low QF=5, and threshold = 0.0005), less error-sensitive blocks are compressed even more, while sensitive blocks are compressed less to achieve higher overall compression ratio at the same accuracy (0.9) [Figure 6.9(b)]. At the same compression ratio, the adaptive approach achieves higher accuracy by preserving weights with higher error sensitivity. By reducing the encoded bits for error-insensitive blocks, the adaptive JPEG encoding method achieves 42X compression

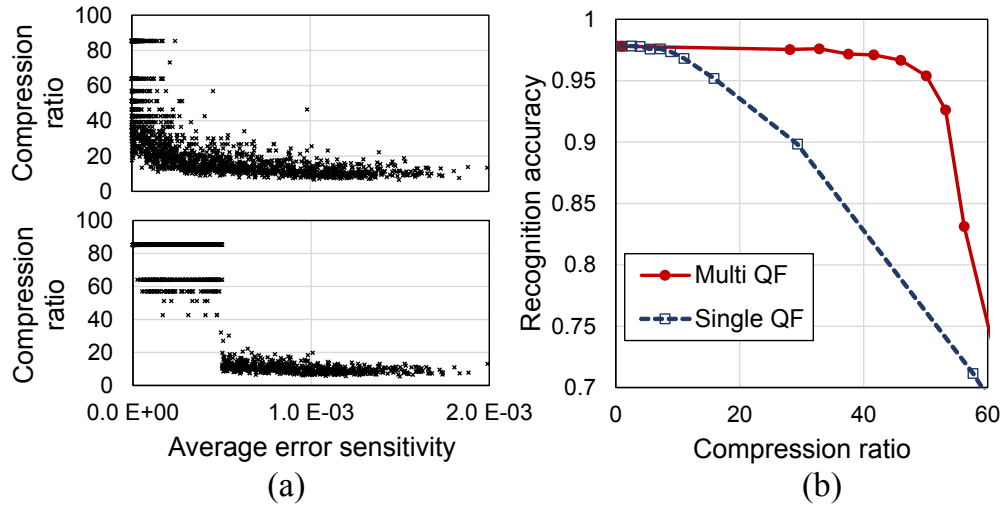


Figure 6.9: (a) Compression ratio vs. average error sensitivity with (top) single QF and (bottom) multi QF. (b) Compression performance with single/multi QF.

(4.3X more compression than the single-QF method) at 1% loss of normalized accuracy [Figure 6.9(c)].

Using more QF values will enable fine-grained compression level control of each block. The fine-grained control is expected to enhance the compression-accuracy performance since QF allocation can better follow the error sensitivity or the entropy distribution of the weight blocks. I apply four QF configurations with different number of QF values (1, 2, 4, 8), to MLP and LeNet-5 for MNIST dataset. As more QF values are used, the QF values are spread between high and low QF values. As Figure 6.10(b) shows, some of the blocks that were encoded with high QF can be encoded with lower QF values, resulting in higher compression ratio. On the other hand, some of the blocks encoded with very low QF can be encoded with higher QF values to preserve more information. Therefore, as Figure 6.10(c) shows, using more QF values increases the compression ratio for a given accuracy level. Although more QF values reduces the compressed size of the weight data, the amount of data that should be stored in memory also includes the header information with the QF values used and the QF allocation map. Assuming QF value range of 1-100, the configuration using n QF values requires 7n bits per matrix for storing the QF values

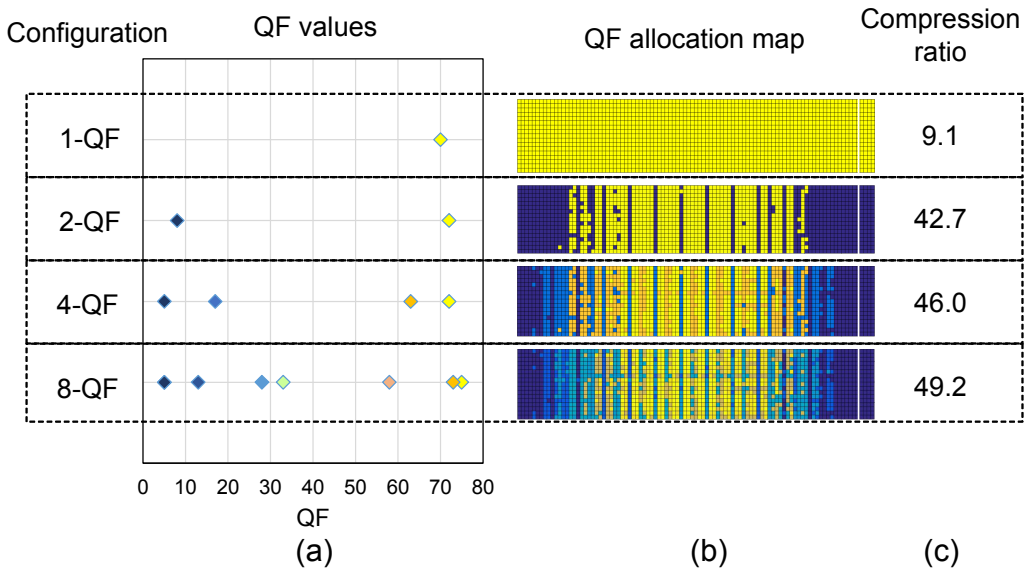


Figure 6.10: (a) Allocated QF values, (b) QF allocation map, and (c) compression ratio of four different QF configurations.

and logn bits per block for QF allocation map. By considering the header size, we can compute the effective compression ratio, the ratio that accounts for the size of the entire data to be stored (compressed weights + header).

Table 6.1 shows that using more QF values increases the compression ratio of the weights, but does not always increase the effective compression ratio because of increased header overhead. Even if we use more than 8 QF values, the original compression ratio will not increase much, while the header overhead will increase linearly. Therefore, we can expect that the effective compression ratio will keep decreasing. Based on this result, using two QFs gives highest effective compression ratio.

1-D and 2-D JPEG Compression Figure 6.11(a) shows the compression performance comparison between the original 2-D and modified 1-D processing on MLP for MNIST. The figure indicates that 1-D processing achieves higher compression ratio than the 2-D processing because it better captures redundancy of 1D weight vector. However, as Figure 6.11(b) shows, the benefit of 1-D processing decreases as the network becomes deeper. The

Number of QFs			1	2	4	8
QF value bits (per matrix)			7	14	28	56
QF map bits (per block)			-	1	2	3
Total overhead (per block)			0.004	1.008	2.02	3.03
Comp. ratio	MLP	Original	9.1	42.7	46.0	49.2
		Effective	9.1	39.4	38.9	38.1
	LeNet-5	Original	21.16	29.6	31.4	32.8
		Effective	21.2	28	27.9	27.5

Table 6.1: Overhead and effective compression ratio of using different number of QFs

reason is that, as the locality reduces, the difference of redundancy along the 1-D and 2-D weight values also decreases.

Gradient and Entropy based QF Selection As Figure 6.12(a) shows, the weight blocks with higher error sensitivity tend to have higher entropy. To compare the actual compression performance, I use both the entropy and the error sensitivity to compress MLP and LeNet-5 for MNIST. The figure also shows that the use of the entropy shows similar compression performance as the use of the error sensitivity. Therefore, to compress given trained weights, the entropy can be a good option that involves less additional process.

Block Formation Method I compare compression performance of the block-wise and column-wise block formation approaches. Figure 6.12(b) shows that the column-wise approach achieves slightly higher accuracy at the same compression ratio. The column-wise encoding benefits from the fact that weights within the same row are more similar.

Summary of the Parameters Based on the above analysis, I determine the parameters that will be used in the rest of the section. First, I use two QF values for adaptive compression as it provides the highest effective compression ratio. For JPEG encoding, 1-D processing is used since it achieves better compression performance with lower computa-

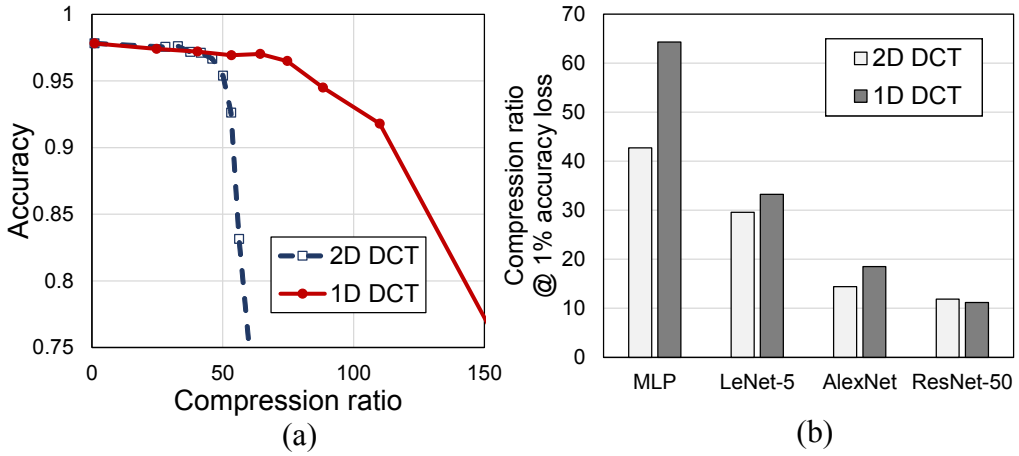


Figure 6.11: Comparison of compression performance of 1-D and 2-D JPEG encoding. (a) Compression ratio vs. recognition accuracy on MLP for MNIST. (b) Compression ratio at 1

tional complexity. Also, entropy-based QF selection is used as it requires less computation than gradient computation with similar compression performance. For JPEG block formation, I use column-wise formation method.

Compression Performance on Multi-Layer Perceptron

For comparison of compression performance I consider four existing compression methods: lossless coding based on run-length encoding and Huffman coding (RLE+Huffman) [52][51], dropout of high frequency components of the weights (DropFreq) [50], pruning [52][47], and truncation [53]. For all compression methods, I do not perform fine-tuning after compression. For the proposed approach, the selection of the high and low QF values are based on a heuristic method. First, I search for the lowest QF value that achieves the target accuracy under the single QF configuration. This QF value is selected as the initial high QF of the 2-QF configuration. Then I search for the low QF value that achieves maximum compression ratio for the target accuracy. If the target accuracy is not satisfied, I increase the high QF value and iterate the search for the low QF value. Figure 6.13 shows compression ratio achieved by different approaches at 1% loss of normalized accuracy. For the

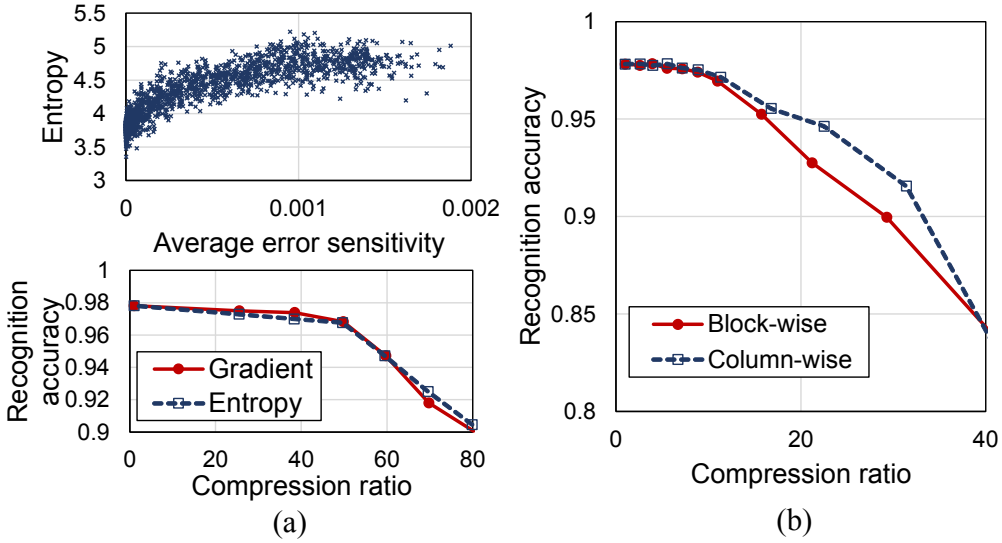


Figure 6.12: (a) Comparison of the error sensitivity and the entropy: (top) Relation between the two and (bottom) compression performance when using the two information. (b) adaptive JPEG compression with block-wise/column-wise block formation approaches.

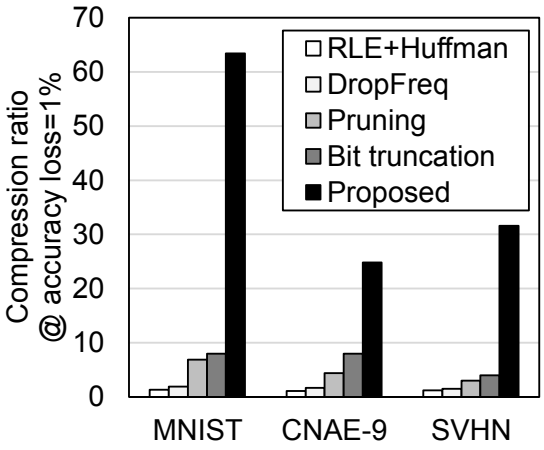
MNIST dataset, the proposed adaptive QF control approach achieves 63.4X compression, which is significantly higher than other approaches. When assuming no accuracy loss, it provides 33X reduction, which is comparable performance to the existing approach with fine tuning[52] (40X reduction on a similar 4-layer MLP).

Compression Performance on CNNs

I apply the proposed compression approach to CNNs that have both convolutional and fully-connected layers. I first analyze the compression performance of fully-connected layers when they are placed deeper in networks. Figure 6.14(a) shows visualization of the weight matrix of the first fully connected layer in a 3-layer MLP and 4-layer LeNet-5 (a CNN), both trained for MNIST dataset. Compared to the image of the MLP weights, the LeNet-5 weights image shows more randomness. This difference is clearly shown in [Figure 6.14(b)], where the weight blocks in LeNet-5 have higher the entropy than that in MLP. The major reason for patterns in the weight image is the locality of input images; some regions of the images are not important for its recognition task, thus the corresponding weight

Dataset	Layer	Original size (KB)	Comp. size (KB)	Comp. ratio
MNIST	FC1	441.0	6.9	64.3
	FC2	5.6	0.2	30.1
	Total	**446.6**	**7.0**	**63.4**
CNAE-9	FC1	481.5	18.2	26.5
	FC2	36.0	2.3	15.4
	FC3	2.3	0.4	5.5
	Total	**519.8**	**21.0**	**24.8**
SVHN	FC1	576.0	13.6	42.3
	FC2	36.0	5.3	6.8
	FC3	2.5	0.5	4.9
	Total	**614.5**	**19.4**	**31.6**

(a)

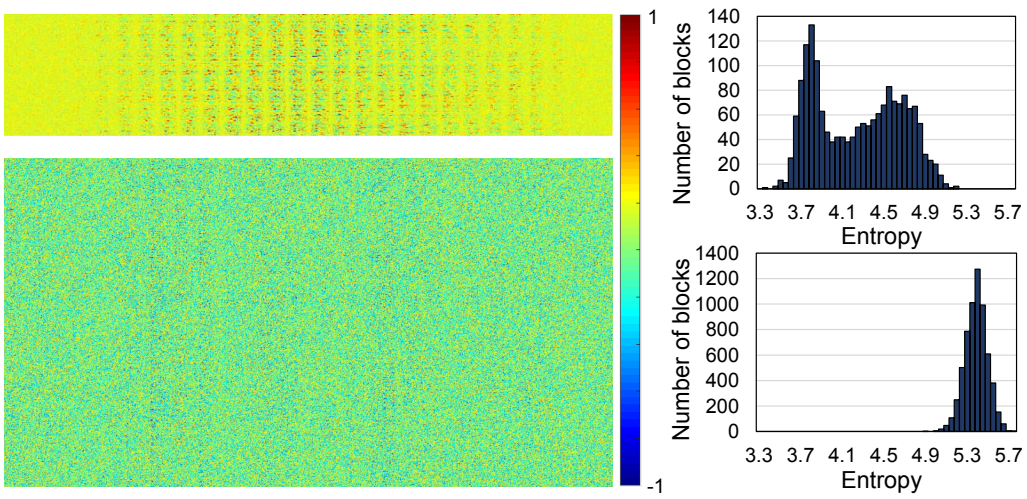


(b)

Figure 6.13: Comparison of compression performance on MLP. (a) Compression ratio of different layers for various datasets. (b) Compression ratio at 1% accuracy loss for various datasets.

elements have less information critical for recognition accuracy. In MLP, the weight matrix for compression is from the first layer, so the locality of the input dataset is preserved in the weight matrix. However, In LeNet-5, two convolution layers are placed before the fully connected layers being compressed. Therefore, the locality of input features is not transferred to the weights, as the features are processed by convolutions, pooling, and activation functions through the network. As a result, information is spread throughout the weight elements in LeNet-5, while in MLP the information is localized to certain regions. Another reason for high entropy of LeNet-5 is that LeNet-5 contains more information than MLP. This can be explained by the fact that the base accuracy of LeNet-5 (0.991) is higher than that of MLP (0.978).

An image with higher entropy implies that it has more randomness and thus more difficult to be compressed. Therefore, the fully connected layer weights of LeNet-5 are expected to have lower compression ratio than the weights in MLP under the same target accuracy. I apply the double-QF adaptive compression approach to the two networks trained for the same MNIST dataset. As expected, LeNet-5 shows lower compression ratio than MLP for the same accuracy [Figure 6.15(a)]. Although lower than MLP, LeNet-5 weights

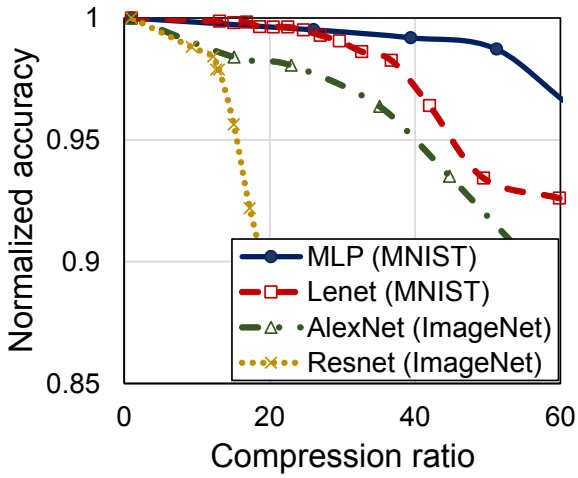


(a) (b)

Figure 6.14: Comparison of weights of MLP (top) and LeNet-5 (bottom). (a) Visualization of a weight matrix, (b) the entropy distribution of weight blocks.

can be compressed by 32.7X at 1% loss of accuracy using the proposed approach [Figure 6.15(b)]. I apply the proposed compression technique to even deeper CNNs, AlexNet and ResNet-50 trained for the ImageNet dataset. Only their fully-connected layer weights are compressed while convolutional layer weights remain uncompressed. AlexNet has five convolutional layers before three fully-connected layers, and ResNet-50 has 49 convolutional layers before one fully-connected layer. Therefore, we can expect that locality of the input dataset is not well transferred to the fully-connected layer weights. Moreover, as the network is trained for much complex dataset, more information will be packed into the weights, leading to higher entropy and lower compression ratio. As expected, Figure 6.15(a) shows that a weight matrix in a deeper network shows lower compression ratio for a target accuracy. However, the proposed approach still provides 18.5X and 11.9X compression for fully-connected layers of AlexNet and ResNet-50, respectively [Figure 6.15(b)].

In Table II, I compare accuracy at certain compression ratio of FC6 and FC7 layers with pruning [52][47], low-rank approximation [48], and the combination of matrix factorization and pruning [55]. Note that all these accuracy values are obtained without retraining in [55]. The table shows that the proposed approach has less accuracy loss than other methods



(a)

Network	Original size (Byte)	Comp. Size (Byte)	Comp. ratio
MLP	446.6 K	7.0 K	63.4
LeNet-5	1,620 K	49.5 K	32.7
AlexNet	224 M	12.1 M	18.5
ResNet-50	8 M	0.7 M	11.9

(b)

Figure 6.15: Comparison of compression performance on fully-connected layers (their convolutional layer weights remain uncompressed). (a) Compression performance of FC layers in different networks. (b) Compression ratio of FC layers at 1

Layer	Size (MB)	Comp. ratio	Pruning	Low-rank approx.	Factoriz. + pruning	Proposed
FC6	144	31	-45.7%	-24.1%	-9.8%	-0.85%
FC7	64	42	-45.7%	-22.7%	-8.0%	-7.55%

Table 6.2: Compression performance (accuracy loss) on two FC layers of AlexNet

for the same compression ratio. In addition to fully-connected layers, I also compress convolutional layers for full-network compression. When applying the proposed approach to compression of the full network with many layers, the optimal set of QF values of each layer should be determined that makes the highest overall compression ratio. To get better perspective of the QF allocation, I first compress each layer with the same set of QF values, and check the compression ratio and accuracy.

Figure 6.16 shows that in both AlexNet and ResNet-50, layers closer to the input are more sensitive to the compression (i.e., more accuracy loss at the same compression ratio). Those layers generally have smaller weight size, so our compression strategy is to compress the earlier layers less with higher QF values, and compress the deeper layers more with

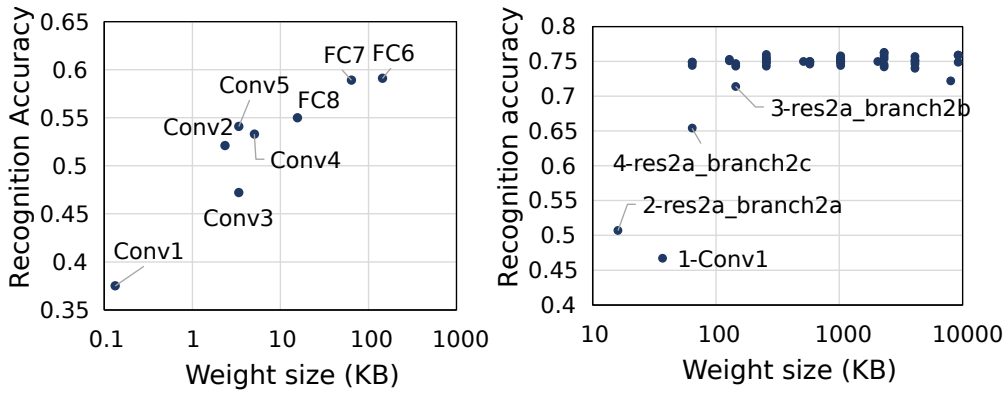


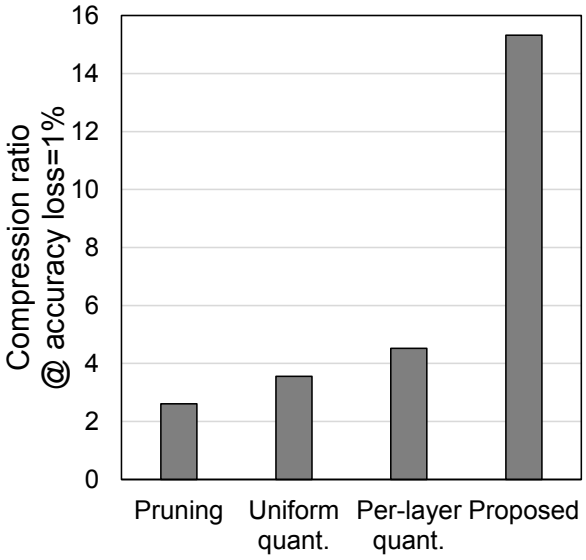
Figure 6.16: Compression performance of each layer in (a) AlexNet and (b) ResNet-50. Each point of the plot represents the original size of each layer weights and the accuracy when only the corresponding layer is compressed into the same target compression ratio (alexnet = 20, resnet = 15), while other layers remain uncompressed.

lower QF values. This approach is applied to the full network compression of AlexNet and ResNet-50. For example, I try to aggressively compress the FC6 layer of AlexNet, which is the largest in size and least sensitive to the compression. On the other hand, its first layer is kept uncompressed.

Figure 6.17(a) shows that the full network compression ratio is 15.3, which is 3.3X more compression than the per-layer quantization method [45] at the same 1% accuracy loss. For ResNet-50, I achieve the full network compression ratio of 10.2. With 32X compression of AlexNet, the accuracy loss is 6.9%, which is higher loss than the existing compression methods that require retraining; the binary-weight network has 2.8% accuracy loss at the same 32X compression [54], and Deep compression achieves 35X compression without impacting accuracy loss [52]. Although these approaches give better compression performance, their major drawback is that they are coupled with the training process during/after compression. Also, they usually require an inference engine with dedicated data flow logic to maximize the benefit from compression. For example, Deep compression stores the weight structure as sparse format encoded with the index difference. This special encoding approach leads to complex and dedicated decoder design especially for reading a sparse matrix representation. In the applications where retraining and redesign of

Layer	Original size (MB)	Comp. size (MB)	Comp. ratio
Conv1	0.13	0.13	1
Conv2	1.17	0.22	5.3
Conv3	3.38	0.55	6.2
Conv4	2.53	0.35	7.2
Conv5	1.69	0.25	6.8
FC6	144	4.90	29.4
FC7	64	7.19	8.9
FC8	16	1.58	10.1
Total	**234**	**15.29**	**15.3**

(a)



(b)

Figure 6.17: Compression performance on AlexNet for ImageNet. (a) Compression ratio of the proposed approach on each layer of AlexNet. (b) Compression ratio of AlexNet at 1% accuracy loss with various approaches.

inference engine is feasible, the methods based on retraining can be a better solution with higher compression ratio. On the other hand, the proposed approach provides a simpler alternative that can be applied to compression of given trained network without retraining or modification of the general inference engine design.

Adaptive Compression with Bit Precision Control

The proposed weight compression technique can be coupled with bit precision control to further enhance the storage/computation efficiency. As Figure 6.18 shows, bit precision control can be inserted at the various stages of the process; the weights can be trained with low precision, or trained weights can be truncated before encoding/after decoding. Note bit precision control cannot be applied to the compressed weights because the compressed weights are stream of Huffman-coded bits, which does not have the concept of bit width.

First, I discuss the case that the LSBs of the trained weights are truncated before supplied to the JPEG encoder. To better understand the effect of bit truncation, I first analyze

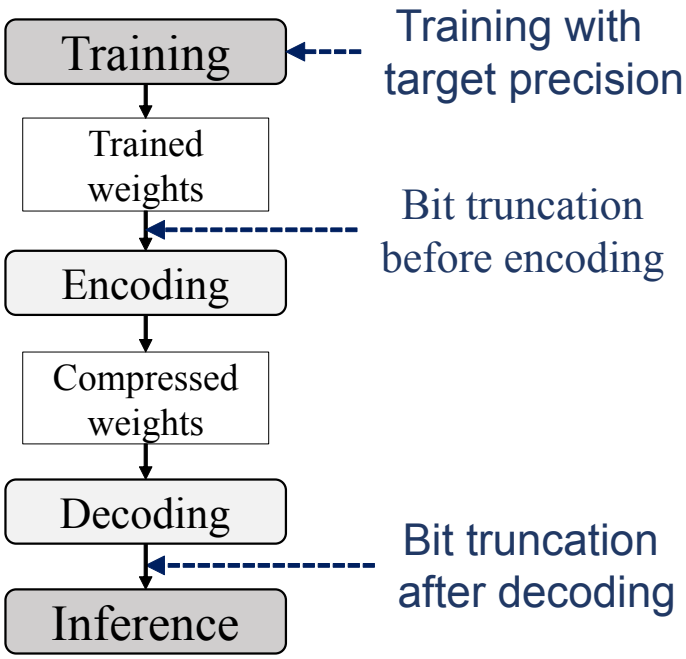


Figure 6.18: Diagram of the network training, weight compression, and inference. Bit precision control can be inserted three stages.

the accuracy when the weights are only truncated (without JPEG encoding). As Figure 6.19(a) shows, the weights with 8 or more bit width show similar accuracy, and the accuracy starts degrading when truncated further into less than 8 bits. However, with JPEG encoding, the weights with 6 bits achieve higher compression ratio than the one with 8 bits as shown in Figure 6.19(b). This is because 6 bit-width weights have smaller set of possible values in a block, resulting in lower the entropy as shown in Figure 6.19(a). Therefore, truncated weights become easier to be compressed, although some information was lost due to truncation.

After the compressed weights are decoded for inference, the bit width of decoded weights can be reduced. Reducing bit precision after decoding does not help reduce the memory demand for storing the weights. Instead, it can help reduce the computation demand and latency of inference by having limited bit precision MAC units. Figure 6.20(a) shows that the accuracy drops as more bits are truncated. The accuracy loss is less than 2% up to 5 bits, but reducing more bits significantly degrades the accuracy. We can min-

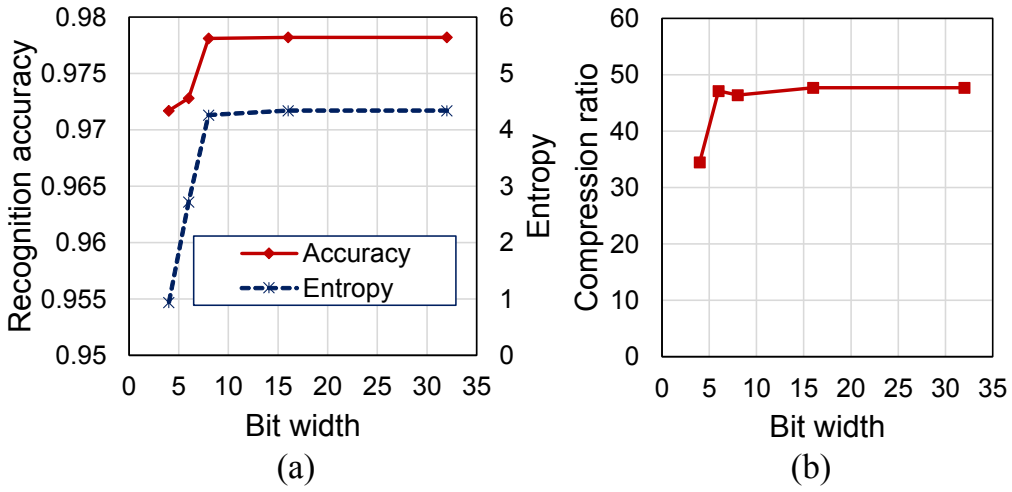


Figure 6.19: Simulation results for bit truncation before encoding. (a) Accuracy and the entropy when the weights are truncated without encoding. (b) Compression ratio of the truncated weights at 1

imize accuracy loss due to bit truncation by less compressing the weights. For example, if one wants to maintain the accuracy after truncating decoded weights, one can apply a set of higher QF values to less compress the weights. As Figure 6.19(b) shows, maintaining accuracy with 4-bit weights degrades the compression ratio from 47x to 36x. If the weights are to be truncated into 3 or less bits, it cannot achieve less than 1% loss of accuracy. This approach will lead to higher memory demand for storing compressed weights, but lower computation demand for inference. Training with low precision reduces latency and computation demand for training as well as memory overhead for weights storage. For simulation, LeNet-5 for MNIST was trained with the full 32-bit precision, and truncated the fully connected layer weights to have 6, 8, 16, and 32 bit widths. I apply fine tuning to make all the bit-width cases have the recognition accuracy of 0.991. The proposed compression technique is applied to the weights trained with four different target bit widths (6, 8, 16, 32). The compression ratio at 1% accuracy loss is shown in Figure 6.20(b). As the figure shows, the weights trained for lower target bit precision can be compressed more at the same accuracy. Major reason for this difference is that training weights for low precision better preserves information than truncating the weights trained for high precision.

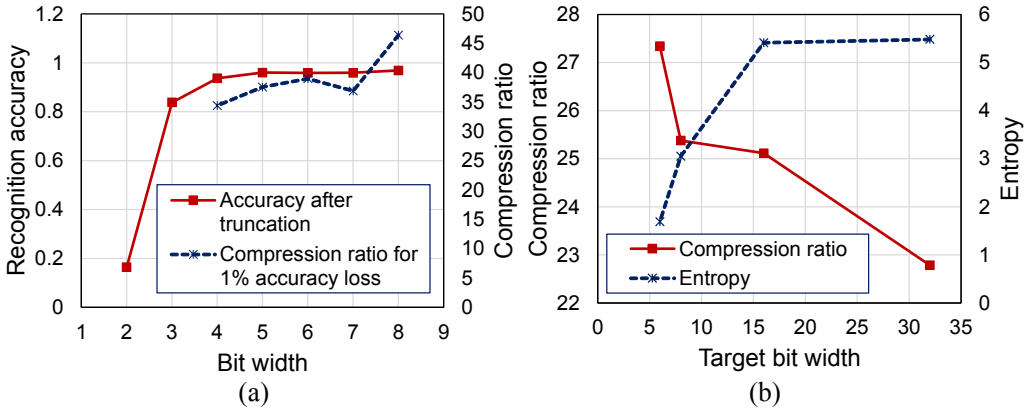


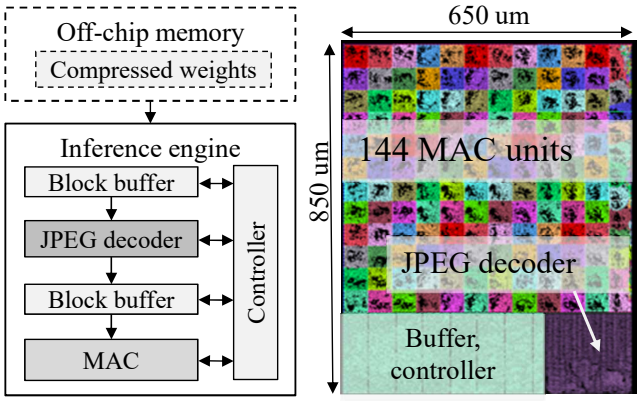
Figure 6.20: Simulation results of bit precision control coupled with weights encoding. (a) Accuracy and compression ratio when bits are truncated after decoding. (b) Compression ratio and the entropy when compressing the weights trained for limited precision.

Another reason is that weights trained for lower precision have smaller set of values in a block. This results in lower the entropy as shown in Figure 6.20(b), making the weights more easily compressed.

6.1.4 System Performance and Energy Analysis

Inference Engine Design

The inference process with the compressed weights at the hardware engine is illustrated in Figure 6.4. First, one block of compressed weights is loaded into the on-chip buffer, together with the QF information. The block is decoded by a JPEG decoder with the same QF used for the encoding. The hardware analysis assumes the standard 2-D JPEG decoder to consider the worst-case decoding overhead. The 1-D JPEG decoder will have less overhead as discussed in Section III.C.4). The decoded weights are fed into a MAC unit for multiplication with input feature. Here, I pipeline the operations (off-chip memory access, JPEG decoding, and MAC operations) in a unit of a block, to maximize the throughput. For system-level performance and energy analysis, I synthesize the engine in 28nm CMOS technology [Figure 6.21]. The on-chip system includes 144 MAC units, a JPEG decoder, and two block buffers with 1 KB for block-level pipelining. The DDR controller is not



(a) (b)

Module	Area (mm^2)	Energy (nJ/block)
JPEG decoder	0.032	1.15
MAC units	0.436	1.03
Buffer/controller	0.085	0.48
Total	**0.553**	**2.66**

(c)

Figure 6.21: Hardware implementation of the inference engine. (a) Block diagram and (b) layout of the engine with decompression and inference modules. (c) Area and energy consumption of each module (energy consumption is for decoding/processing one block of compressed weights).

included in the on-chip system. As the inference engine was originally designed for processing the MLP for MNIST, 144 MACs are chosen to process 144 rows of the MNIST weights at a time. The 144 MACs can also be configured to process multiple elements in a column. The operating frequency is 1 GHz, and the power and latency numbers are obtained from post-layout simulation. I also assume the inference batch size of 1 (i.e. a single input image is supplied at a time).

System Performance and Energy Analysis

As compression reduces the memory size required to store the trained weights, memory access delay and energy during inference will decrease accordingly. However, the proposed compression approach requires JPEG decoding, which incurs additional computation time and energy. Therefore, to achieve the advantage in system throughput and energy, compression ratio should be high enough so that the reduction in memory access time and energy can compensate for decoding overhead. As the memory access time and energy vary depending on the off-chip memory type, I performed the throughput and energy estimation

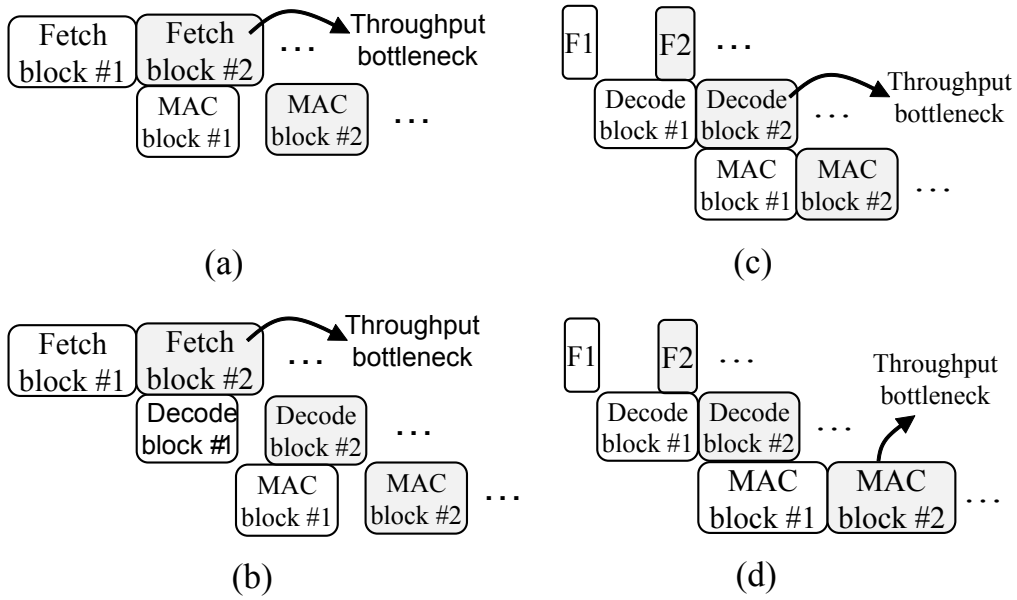


Figure 6.22: (a) Pipelining diagram without JPEG decoding. (b) Pipelining diagram with JPEG decoding when the bottleneck is in the stage of (b) memory access, (c) decoding, and (d) MAC operations.

using the specifications of two different memory types: DDR3 [114] and SD card [115]. The reason for considering SD card, a non-volatile memory (NVM), is to realize the fact that if the weights are not updated regularly on-line, storing the weights in NVM can significantly reduce the storage energy. However, SD cards have higher access time/energy than DDR3. The access latency and energy of DDR3 are 0.19 ns/bit, and 70 pJ/bit [125], and 10 ns/bit and 3600 pJ/bit for SD card [115]. The bandwidth of DDR3 and SD card is 12.8 GB/s [126] and 12.5 MB/s [127], respectively. The memory demand for the datasets presented here can in principle be stored in an on-chip memory as they are based on relatively small images. However, as discussed in Figure 6.1, even for the same applications more practical image sizes will make on-chip storage impossible. Therefore, for our analysis I have considered an off-chip storage for all applications, with and without compression.

I first analyze the effective memory throughput (performance) between the off-chip memory and MAC units, indicating how many synaptic weights can be supplied to MAC units per unit time (GWPS: Giga weights per second). In the baseline system without com-

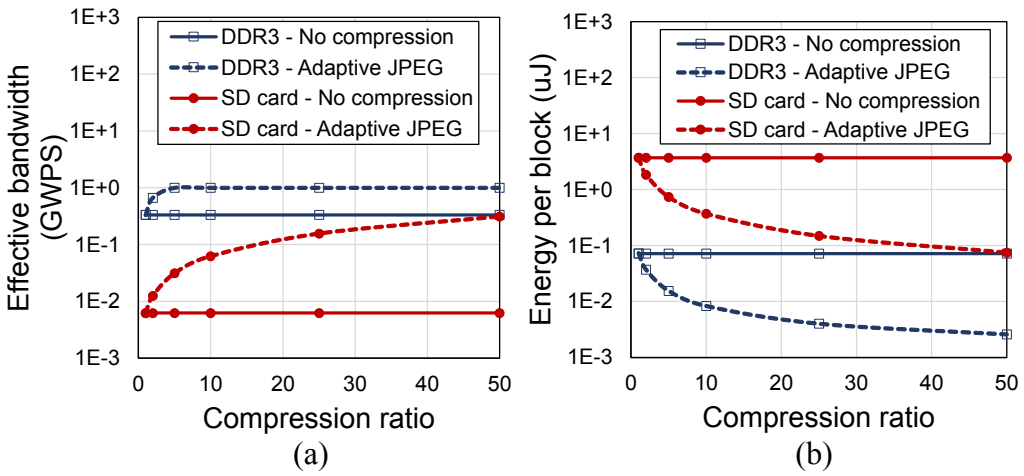


Figure 6.23: (a) Effective bandwidth and (b) energy per block with compression ratio.

pression, the effective bandwidth is limited by the off-chip memory access time of a block [Figure 6.22(a)]. As access time for a block decreases due to the adaptive JPEG compression, the effective memory throughput increases even considering the decoding time [Figure 6.22(b) and Figure 6.22(a)]. Note that pipelining fetch and JPEG decoding stages helps improve the performance. As compression ratio increases, reduced block access time enhances the throughput until the decoding time becomes the throughput bottleneck [Figure 6.22(c)]. Therefore, the system throughput saturates at higher compression as illustrated in Figure 6.23(a). The effect is more pronounced for DDR3 with higher memory performance, compared to the SD card case. To analyze the energy advantage, I illustrate the sum of memory access and decoding energy for given compression ratio [Figure 6.23(b)]. For all memory types, compression leads to significant reduction in energy because of reduced memory access energy.

Table 6.3 shows the memory throughput and system energy of the proposed approach and pruning for the MNIST dataset, and Table 6.4 compares the throughput and energy improvement for different networks, datasets, and off-chip memory types. For relatively lower latency memory (DDR3), high compression ratio of the proposed method does not translate into effective throughout improvement due to decoding time overhead. However,

Off-chip memory		DDR3			SD card		
Compression		No	Pruning	Adaptive JPEG	No	Pruning	Adaptive JPEG
Compression ratio		1	5.9	**63.4**	1	5.9	**63.4**
Throughput (GWPS)		0.33	1.95	1	0.006	0.037	0.40
Throughput improvement		-	**5.9**	3	-	5.9	**63.4**
Energy per block (nJ)	Off-chip mem	71.7	12.1	1.13	3686	624.9	58.1
	Decoder	-	-	1.15	-	-	1.15
	MAC	1.03	1.03	1.03	1.03	1.03	1.03
	Total	72.7	13.2	3.31	3687	626	60.3
Energy reduction		-	5.5	**22.0**	-	5.9	**61.1**

Table 6.3: Memory throughput and system energy comparison

Off-chip memory	Network (Dataset)	Compression	Comp. ratio	Throughput improvement	Energy reduction
DDR3	MLP (MNIST)	Pruning	5.9	**5.9**	5.5
		Adaptive JPEG	**63.4**	3	**22.0**
	AlexNet (ImageNet)	Pruning	2.6	1.8	1.8
		Adaptive JPEG	**15.3**	**2.7**	**1.8**
SD card	MLP (MNIST)	Pruning	5.9	5.9	5.9
		Adaptive JPEG	**63.4**	**63.4**	**60.8**
	AlexNet (ImageNet)	Pruning	2.6	1.7	2.5
		Adaptive JPEG	**15.3**	**6.8**	**3.7**

Table 6.4: Comparison of throughput and energy improvement

when the off-chip memory access latency is significantly larger than decoding (SD card), the proposed approach achieves much higher throughput improvement than pruning, with its high compression ratio. Also, its higher compression results in significant reduction in system energy even with decoding energy overhead. The energy-efficiency improvement becomes higher when a system uses SD card, where the system energy is dominated by access energy.

Table 6.5 shows the energy and throughput improvements by the proposed compression technique, for various networks. Here I consider the throughput improvement and energy reduction in both memory and system perspectives. Memory throughput/energy indicates efficiency of data movement from off-chip memory and MAC units, and includes only off-chip memory access and weight decoding. System throughput/energy indicates efficiency

Network	Comp. ratio	DDR3				SD card			
		Throughput improvement		Energy reduction		Throughput improvement		Energy reduction	
		Mem	Sys	Mem	Sys	Mem	Sys	Mem	Sys
MLP	63.4	3	3	31.0	22.0	63.4	63.4	62.2	61.1
LeNet-5	31.3	3	3	21.2	3.8	31.3	9.8	30.9	8.3
AlexNet	15.3	2.7	2.7	12.2	1.8	15.3	6.8	15.2	3.7
ResNet-50	10.2	2.4	2.4	8.8	1.3	10.2	4.2	10.1	2.5

Table 6.5: Throughput and energy improvement for different networks

of end-to-end inference by considering computation energy/latency of the MAC units as well. For energy/throughput estimation of convolutional layers, I assume partial kernels are sequentially loaded/decoded for convolution with input feature maps. As 64 elements of the kernel along the channel and depth direction are compressed in a block, partial kernels with dimension of k k 64 are loaded and decoded for convolution with corresponding part of input feature map to generate a partial output feature. The table shows that, the advantage is limited with faster and low-power memory (DDR3) due to the decoder overhead. On the other hand, with a SD card, compression ratio of each network is translated into the improvements because the system energy and throughput are dominated by memory access energy and latency. Compared to the MLP, the compression ratio of CNNs is not fully translated to the energy reduction because MAC operations dominate the CNN energy consumption.

Analysis with On-Chip Storage of Weights

System analysis presented above assumes the compressed weights are stored in an off-chip memory of the inference engine. I have shown that the adaptive weight compression scheme reduces the off-chip memory access energy and latency, hence, enhances the overall system energy efficiency and throughput. If the compressed weights fit into the on-chip

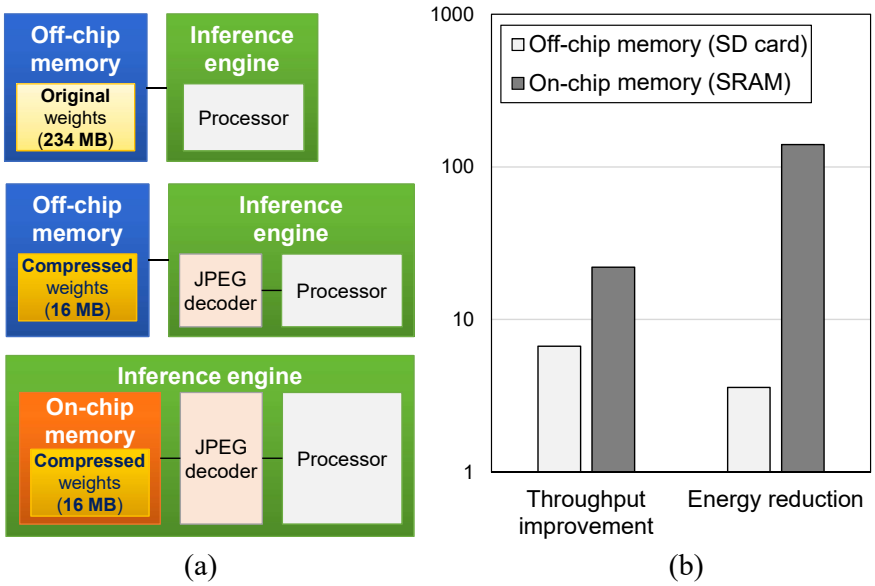


Figure 6.24: (a) Three different configurations of an inference engine. (Top) Uncompressed weights stored in off-chip memory, (middle) compressed weights stored in off-chip memory, and (bottom) compressed weights stored in on-chip memory (b) Throughput and energy improvement with off-chip and on-chip memory.

memory of the inference engine, we can store the weights in the on-chip memory which generally has lower access energy and latency. Figure 6.24(a) shows three different storage configurations of the inference system with AlexNet for ImageNet dataset. The baseline configuration is to store uncompressed weights in an off-chip SD card. By compressing the weights, smaller weights are stored in an off-chip memory with a slight overhead of JPEG decoder. As the weights can be compressed into 16MB through the proposed technique, it can fit into on-chip memory of recent processors. As Figure 6.24(b) shows, storing compressed weights in an on-chip memory further improves energy efficiency and throughput.

6.1.5 Summary of the Section

This section presented an energy efficient design of a neural network inference engine based on the adaptive weights compression using JPEG image encoding. By adaptively determining the quantization level of JPEG, the proposed approach achieves high compression ratio

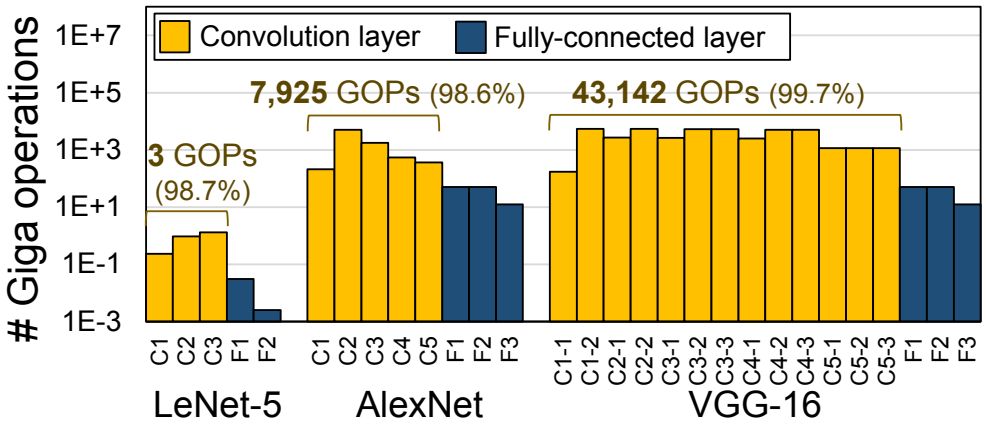


Figure 6.25: Computational complexity of various CNNs.

while preserving error-sensitive information. The high compression leads to the throughput improvement of the inference hardware system at significantly lower energy even with accounting for the JPEG decoding overhead. To further minimize the overhead, I propose the use of the entropy for the QF allocation, instead of error sensitivity computation. I also explore the optimization of the design by considering the number of QF values, bit truncation techniques, and block formation methods. With reduced memory and computation overhead, the proposed design can be a simple solution for inference with a trained network using resource-constrained hardware.

6.2 Frequency Domain Convolution for Computationally-Efficient DNNs

6.2.1 Introduction

The previous section explored approaches to compress the weights of DNNs for memory-efficient deep learning. However, their high computational complexity is also a major bottleneck, especially for convolutional neural networks (CNNs) have been widely applied in computer vision tasks including image classification [91] and face recognition [92]. For example, 3-convolution-layer LeNet-5 requires 3 Giga operations (GOPs) to train a 32x32 image, while 13 convolutional layer VGG-16 learns a 224x224 image with 43,142

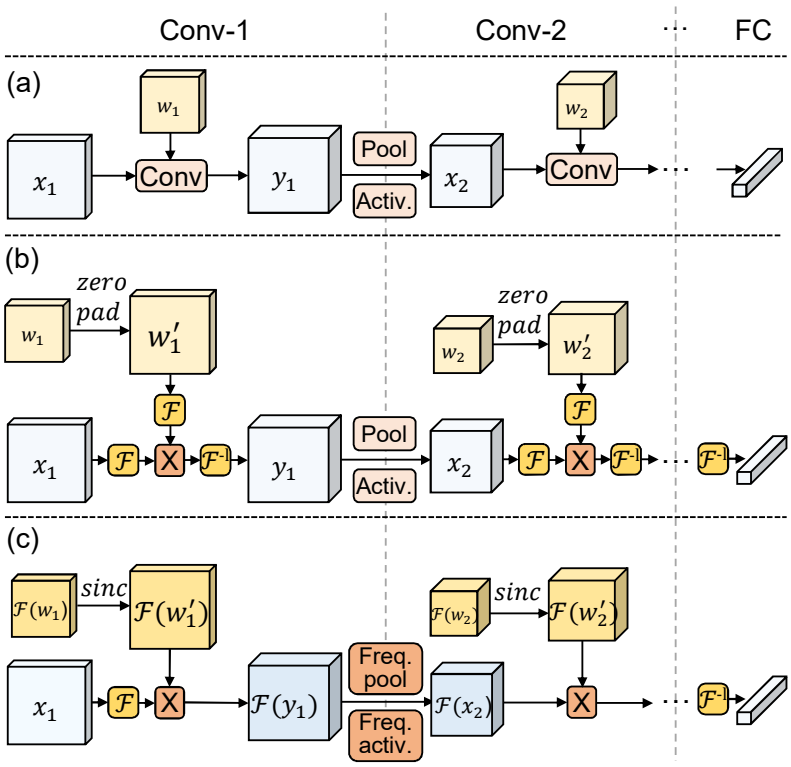


Figure 6.26: CNN models in forward propagation with (a) conventional spatial-domain approach, (b) FFT-based approach, and (c) the proposed entire frequency-domain approach

GOPs [Figure 6.25]. The high computational complexity is a major challenge in training CNNs, particularly, in embedded platforms. Therefore, reducing the computation demand of CNNs is critical, specifically, to support in-field and on-chip learning.

In this section, I propose an accelerator design that maps the entire convolution layer operations and parameters into the frequency domain for fast and energy-efficient CNN training [Figure. 6.26(c)]. The key contributions of this work are:

- I introduce frequency-domain activation functions for the non-linear operations (e.g. pooling, activation function) between convolution layers. Hence, the entire net-

work can be trained with frequency-domain computation without the need for the FFTs/IFFTs in-between the layers.

- I propose to store frequency-domain kernels without zero padding to eliminate memory overhead, and expand them before computation through sinc interpolation, which is the frequency-domain duality of zero-padding.

- Only half of the parameters are stored and multiplied using Hermitian symmetry to reduce computation and memory.

I design a digital accelerator for training and inference of CNNs with the proposed approach. The simulations with the MNIST and CIFAR-10 datasets show that the proposed approach with the approximate nonlinear operations achieves the same accuracy as the spatial-domain CNN training model. The design is synthesized in FPGA, as well as in 28nm CMOS technology for performance and energy analysis. The results show that the proposed accelerator significantly reduces training time and energy. The advantage becomes larger for a network with larger input and kernel sizes. For AlexNet, I reduce 4.7X and 6.3X energy and latency than the conventional spatial-domain training.

6.2.2 Background

With frequency-domain representations of the parameters, convolutions can be performed through simple element-wise multiplications (FFT-based convolution) [58][59]. Let us assume each layer has f' different kernels of depth f, and a set of f input feature maps, which is a part of minibatch S. During forward propagation, each output feature map $y_{s,j}$ is computed as a sum of the input feature maps $x_{s,i}$ cross-correlated with the corresponding kernel $w_{j,i}$, where $j \in f'$, $i \in f$, $s \in S$. Cross-correlation can be obtained as the inverse Fourier transform of the pointwise product between the individual transforms:

$$y_{s,j} = \sum_{i \in f} x_{s,i} \star w_{j,i} = \sum_{i \in f} \mathcal{F}^{-1} \left(\mathcal{F} \left(x_{s,i} \right) \cdot \mathcal{F} \left(w_{j,i}{}' \right)^{*} \right) \tag{6.4}$$

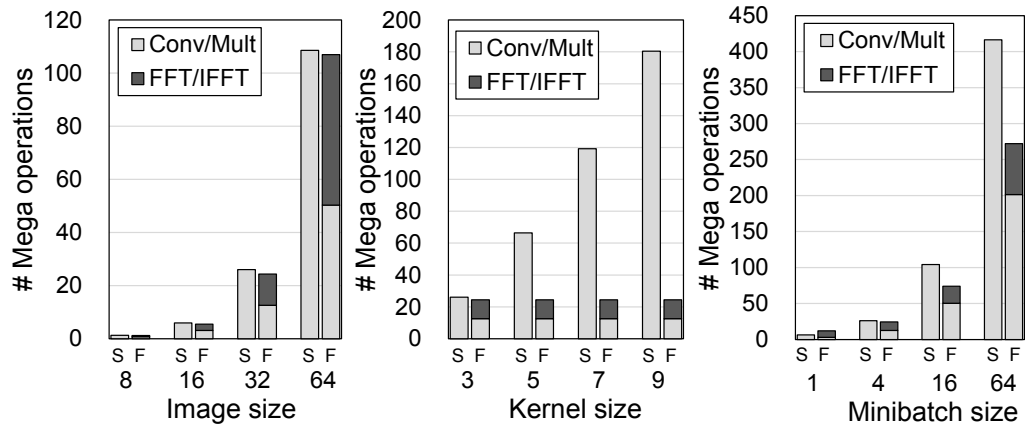


Figure 6.27: Complexity comparison of the spatial-domain approach (S) and FFT-based approach (F) (Default values: input feature size=32x32 and depth=16, kernel size=3x3 and depth=16, and minibatch=4).

where F and F^{-1} are FFT and IFFT, and * denotes complex conjugation. Note that the kernels should be zero-padded ($w_{j,i}{}'$) to be the same size as the input feature before applying the FFT, in order to perform desired linear convolution. During backward propagation, the gradients of loss with respect to the outputs ($\partial L/\partial y_{s,j}$) are convolved with kernels, producing the gradients with respect to the inputs ($\partial L/\partial x_{s,i}$). As convolutions can be replaced with multiplications, we have:

$$\frac{\partial L}{\partial x_{s,i}} = \sum_{j \in f'} \frac{\partial L}{\partial y_{s,j}} * w_{j,i} = \sum_{j \in f'} \mathcal{F}^{-1}\left(\mathcal{F}\left(\frac{\partial L}{\partial y_{s,j}}\right) \cdot \mathcal{F}\left(w_{j,i}{}'\right)\right) \tag{6.5}$$

To calculate the gradients of the loss with respect to the weight ($\partial L/\partial w_{j,i}$), the input feature maps are cross-correlated with the gradients with respect to the outputs. In the frequency domain, it can be calculated by multiplying the two matrices:

$$\frac{\partial L}{\partial w_{j,i}} = \sum_{s \in S} \frac{\partial L}{\partial y_{s,j}} \star x_{s,i} = \sum_{s \in S} \mathcal{F}^{-1}(\mathcal{F}\left(\frac{\partial L}{\partial y_{s,j}}\right) \cdot \mathcal{F}\left(x_{s,i}\right)^*) \tag{6.6}$$

In this approach, every convolution involves three Fourier transforms. As some of the Fourier representations obtained in the forward pass are used in the backward pass, Mathieu et al. [58] proposed storing and reusing them to reduce the transform overhead. However,

the complexity comparison with the spatial-domain convolution [Figure 6.27] shows that the transform overhead still accounts for large portion of the total complexity, making it less efficient than spatial convolution for small kernel and minibatch sizes. Furthermore, this approach requires significant amount of additional memory to store the Fourier representations. In this approach the transforms could not be removed because the output feature maps should be transformed back to the spatial domain for non-linear operations (e.g. pooling, activation function), which do not have exact dual operations in the frequency domain. In this work, I use approximate frequency-domain nonlinear operations that enable us to maintain the feature maps in the frequency domain after convolutions. Kernels are also initialized and updated directly in the frequency domain, thereby eliminating the Fourier transforms inside the entire set of convolution layers. To avoid memory increase, I exploit sinc interpolation and subsampling techniques that allow us to store the frequency-domain kernels with the original dimension. Using Hermitian symmetry of the parameters, I further reduce computation and memory demand by storing and computing only part of the parameters.

6.2.3 Proposed Approach

Overall Structure

As Figure 6.28 illustrates, the proposed training approach uses the same processes as the conventional spatial-domain CNN model, but with the frequency-domain parameters and operations. At the beginning of the forward pass, a set of input images is transformed into their Fourier representations [Figure 6.28(a)]. Note that the input images can be prepared as the Fourier representations through the off-line transforms before training in order to reduce the on-line training time. Frequency-domain kernels can be initialized by transforming the spatial-domain initial weights, which can be generated by various initialization techniques. However, this requires the transform overhead for the entire kernel sets, although the transforms are performed only once at the beginning of the training. Alternatively, the weights

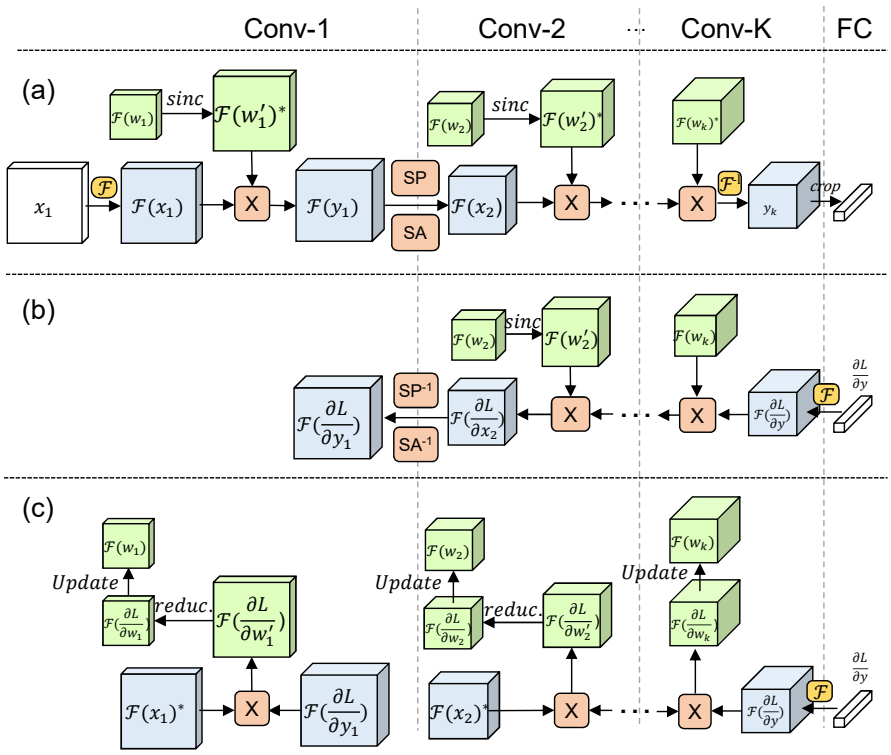


Figure 6.28: CNN model of the proposed approach in (a) forward propagation, (b) backward propagation, and (c) gradient calculation and update (SP, SA: spectral pooling/activation). Note that no FFT is required in (b) and (c) if L/y from the fully-connected layer is 1x1 shape.

can be initialized directly in the frequency domain by randomly setting the complex numbers using the initialization techniques. Note that kernels are initialized and stored without zero-padding ($\mathcal{F}(w)$). They are expanded through sinc interpolation ($\mathcal{F}(w\ ')$) before being multiplied with the input feature maps during forward propagation. The output feature maps are then processed through pooling and activation in the frequency domain. Originally the input feature map dimension $(n \times n)$ reduces to $(n-k+1) \times (n-k+1)$ after convolution with kk kernel, and then reduces to half $((n-k+1)/2 \times (n-k+1)/2)$ by the pooling operation. However, in the proposed approach, the output dimension after convolution is kept same as the input feature maps to avoid the overhead of the transform and

cropping. Instead, the nn output features are directly reduced to $(n-k+1)/2 \times (n-k+1)/2$ dimension by the spectral pooling that supports arbitrary reduction ratio. Feature maps remain in the frequency domain until the last convolution layer, where they are transformed back into the spatial domain and cropped for the desired dimension. Therefore, the proposed approach requires only two Fourier transforms, which are at the boundaries of the entire set of convolution layers. The entire backward propagation and gradient calculation process is also performed in the frequency domain [Figure 6.28(b-c)]. As the gradients of loss from the first fully-connected layer ($\partial L/\partial y$) is in the spatial domain, they should be transformed into the frequency domain ($\mathcal{F}(\partial L/\partial y)$) for the backward pass towards convolution layers. However, if the gradients ($\partial L/\partial y$) are a set of scalar (1x1 shape) values, their FFT is not necessary since the FFT of scalar value is an identity function. In backward propagation [Figure 6.28(b)], the gradients with respect to the output ($\mathcal{F}(\partial L/\partial y)$) are obtained as the Fourier representation by using the frequency-domain backward pooling and activation function. They are then multiplied with the conjugated input feature maps ($\mathcal{F}(x)^*$) to get the gradients with respect to the weights ($\mathcal{F}(\partial L/\partial w\ ')$), which are used to update frequency-domain kernels [Figure 6.28(c)]. Since the gradients are in the zero-padded dimension, their dimension should be reduced ($\mathcal{F}(\partial L/\partial w)$ in order to update kernels in the original non-zero-padded dimension.

Application of Sinc Interpolation

In order to implement desired linear convolution through the FFT, kk kernels should be zero-padded before the FFT, making their dimension as same as the features $(n \times n)$. Therefore, one may consider storing kernels as the Fourier representation of zero-padded weights. However, considering recent networks with large input image size, this approach will lead to significant increase (n^2/k^2) in memory size and bandwidth.

To avoid this memory increase, I propose storing the FFTs of the original kernels ($\mathcal{F}(w)$) and expanding them to be the FFTs of zero-padded kernels ($\mathcal{F}(w\ ')$) before the

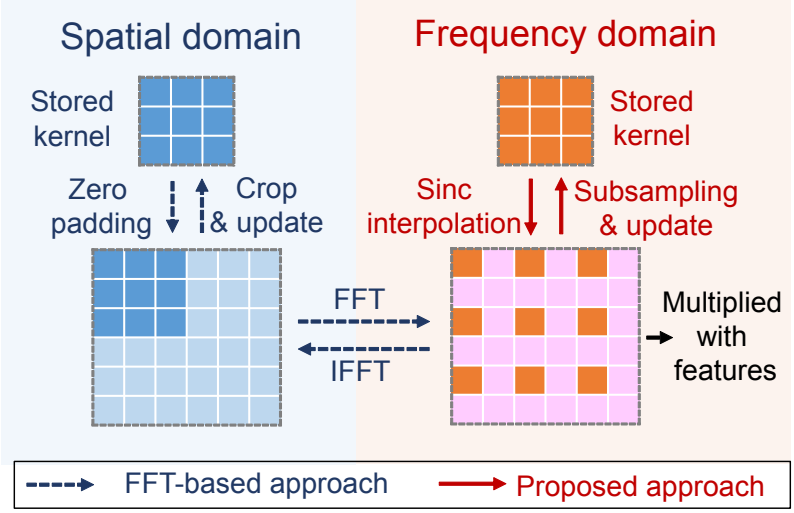


Figure 6.29: Manipulation of kernel in the FFT-based and proposed approach.

computation [Figure 6.29]. For the expansion of kernels, I exploit the zero-padding theorem, stating that the FFT of zero-padded signal can be obtained by convolving the FFT of the original signal with a discrete sinc kernel [7]:

$$G(u,v) = \bar{F}(u,v) * \left[e^{\frac{-j2\pi u}{n}\left(\frac{k-1}{2}\right)} \frac{\sin\left(\frac{\pi u k}{n}\right)}{\sin\left(\frac{\pi u}{n}\right)} \cdot e^{\frac{-j2\pi v}{n}\left(\frac{k-1}{2}\right)} \frac{\sin\left(\frac{\pi v k}{n}\right)}{\sin\left(\frac{\pi v}{n}\right)} \right] \tag{6.7}$$

When updating kernels in the backward pass, the gradients $\mathcal{F}(\partial L/\partial w\,')$ should be reduced to $\mathcal{F}(\partial L/\partial w)$. Generally, the reduction can also be performed through interpolation. However, when the dimension of $\mathcal{F}(w\,')$ is multiple of $\mathcal{F}(w)$ dimension, we can obtain all the values of $\mathcal{F}(w)$ immediately by subsampling the components in $\mathcal{F}(w\,')$. Therefore, with a constraint that the dimension ratio of the input features and kernels in a layer is multiples, no sinc interpolation is required during the gradient calculation of the layer. Sinc interpolation from kk to nn signal requires k^2n^2 complex multiplications, while memory requirement reduces from O (n^2) to O (k^2). Note that no interpolation is needed at the last convolution layer if the shapes of the input feature maps and kernels are same (n=k).

Application of Hermitian Symmetry

Given an nn real-valued signal x, its Fourier transform X has Hermitian (conjugate) symmetry, where negative frequency coefficients are conjugate of positive ones

$$X\left(k,l\right) = \ X^{*}\left(-k \bmod n,\ -l \bmod n\right) \tag{6.8}$$

This property is commonly used to describe the Fourier representation only with the positive-frequency samples. Since the parameters in general CNNs are real-valued, a prior study [58] has exploited this property to represent the FFT of $n \times n$ parameters using $n/2(n+1)$ components with real and imaginary parts. However, some of these components are real values, which do not need to be stored with the two parts. More precisely, FFT of nn real signal has $(n^2 - \alpha)/2$ complex and α real components, where is 1 for odd n and 4 for even n. Therefore, the memory requirement can be further optimized to n^2 real components, the same space as the original real signal. Note that this approach also allows us to perform half the computation by multiplying only the positive-frequency components.

Figure 6.30 shows how this property is exploited in the proposed approach. At the beginning of forward propagation, Fourier-transformed kernels and input features are compressed by discarding negative-frequency components. The positive-frequency components in the compressed kernels are interpolated and multiplied with the corresponding components in the compressed feature maps. The original form of feature maps with negative-frequency components is reconstructed simply by taking the complex conjugate of positive-frequency values before being transformed into the spatial domain at the last convolution layer. Using this symmetry, n^2 complex multiplications reduce to $(n^2 - \alpha)/2$ complex and α real multiplications, where α is 1 for odd n and 4 for even n. For energy-efficient complex multiplication, I use a well-known technique that uses three real products per multiplication [128]. Therefore, for convolution of nn feature and kk kernel, the pro-

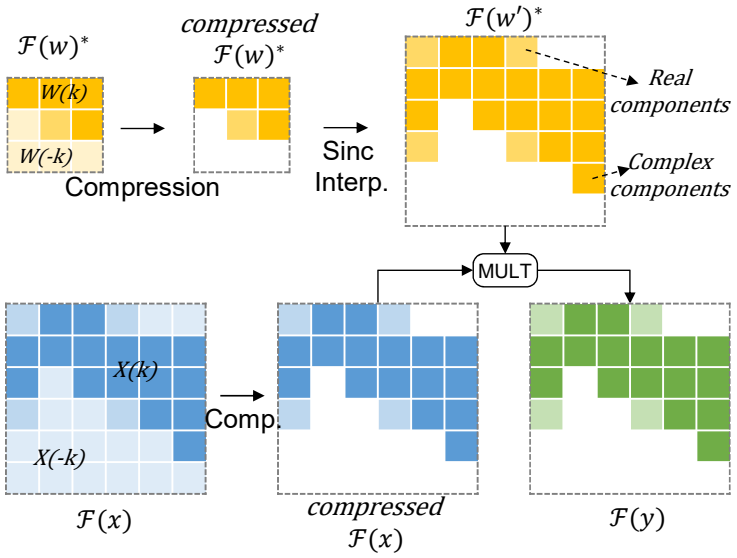


Figure 6.30: Convolution of the compressed kernel and feature map in forward propagation.

posed approach requires $3/2(n^2-\alpha)+\alpha$ real products, while the spatial approach requires $(n-k+1)^2k^2$ real products.

The computational complexity in the backward pass is further reduced in conjunction with the property of zero-padding [Figure 6.31]. As explained in the Section 3.2, the gradients with respect to zero-padded weights ($\mathcal{F}\left(\partial L/\partial w'\right)$) are reduced to the gradients without zero padding ($\mathcal{F}\left(\partial L/\partial w\right)$) by sampling only certain components. Therefore, only the corresponding components in $\mathcal{F}(x)^*$ and $\mathcal{F}(\partial L/\partial y)$ can be multiplied to obtain the gradients. This approach reduces complexity of gradient calculation from $3/2(n^2-\alpha)+\alpha$ to $3/2(k^2-\alpha)+\alpha$, where n, k are the sizes of the input features and kernels, respectively.

Frequency-Domain Nonlinear Operations

For a frequency-domain pooling operation, I use the spectral pooling proposed by Rippel et al. [129]. Given an $\alpha \times \alpha$ frequency-domain input feature map with DC component

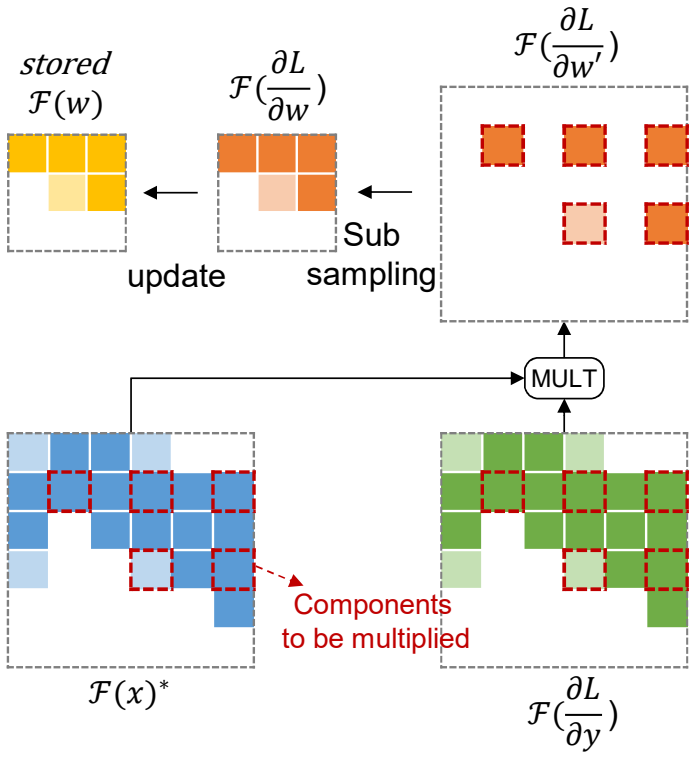


Figure 6.31: Convolution of the output gradients and the input feature maps for obtaining gradients, which are subsampled to update the stored kernels in the backward pass.

shifted to the center, central submatrix with the desired size $\beta \times \beta$ is cropped. The spectral pooling is claimed to be more beneficial than the spatial-domain max or average pooling, since it maintains more information and also allows any arbitrary output dimensionality.

For a frequency-domain activation function, I introduce approximate sigmoid and tanh functions using linearity of FFT. Generally, the input features to the activation function are normalized to have zero means and unit variances to speedup training [130]. This motivates us to focus on the central region of the function, which can be approximated by linear slope as shown in Figure 6.32. In the frequency domain, the linear mapping is easily realized by adding scalar to DC value (shift) or multiplying scalar to all the components (scaling). To

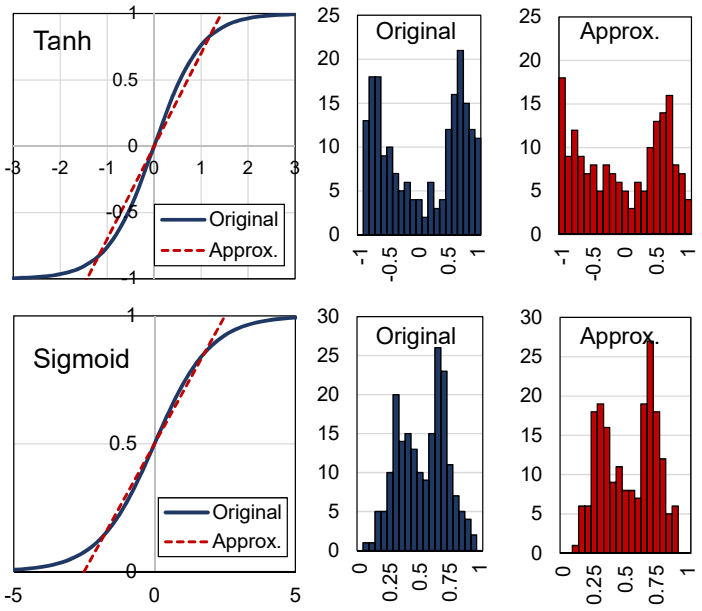


Figure 6.32: (Left): Approximation of activation functions (tanh, sigmoid) in the frequency domain. (Right): Comparison of output feature distribution.

validate the approximation, I apply the original and approximate functions to one of the first convolution layer feature maps of a trained LeNet-5 network for CIFAR-10. Figure 6.32 shows that the distribution after the approximate functions fairly match with the ones with the original functions.

6.2.4 Algorithmic Analysis

Computation and Memory

The complexity of a given convolution layer in terms of the number of multiplication and memory requirement is summarized in Table 6.6. By reducing both the number of Fourier transforms and the number of multiplications for convolution, the proposed approach requires least computation among the three approaches at any size of input feature, kernel,

Approach	Operation	Computational complexity (# real-number multiplication)			Memory requirement (# real number)		
		Inference	Back propagation	Gradient calculation	Weights	Feature	Additional memory for FFT reuse
Spatial	CONV	$S \cdot f' \cdot f \cdot (n-k+1)^2 \cdot k^2$	$S \cdot f' \cdot f \cdot n^2 \cdot k^2$	$S \cdot f' \cdot f \cdot (n-k+1)^2 \cdot k^2$	$f' \cdot f \cdot k^2$	$S \cdot f \cdot n^2$	-
FFT-based	FFT/IFFT	$2Cn^2 \log n \, (S \cdot f + f' \cdot f + f' \cdot S)$	$2Cn^2 \log n \, (S \cdot f + f' \cdot S)$	$2Cn^2 \log n \, (f' \cdot f)$	$f' \cdot f \cdot k^2$	$S \cdot f \cdot n^2$	$n \cdot (n+1) \cdot (S \cdot f + f' \cdot f + f' \cdot S)$
	MULT	$4n^2 \cdot S \cdot f' \cdot f$	$4n^2 \cdot S \cdot f' \cdot f$	$4n^2 \cdot S \cdot f' \cdot f$			
Proposed	FFT/IFFT	$2Cn^2 \log n \, (S \cdot f)$ *(Only at the 1st and last layer)*	-	-	$f' \cdot f \cdot k^2$	$S \cdot f \cdot n^2$	-
	Interpolation/ reduction	$3/2 \cdot n^2 \cdot k^2 \cdot f' \cdot f$	$3/2 \cdot n^2 \cdot k^2 \cdot f' \cdot f$ *(only when $n \neq k$)*	$3/2 \cdot n^2 \cdot k^2 \cdot f' \cdot f$ *(only when $n \neq a \cdot k$)*			
	MULT	$(3/2 \cdot (n^2-\alpha)+\alpha) \cdot S \cdot f' \cdot f$	$(3/2 \cdot (n^2-\alpha)+\alpha) \cdot S \cdot f' \cdot f$	$(3/2 \cdot (k^2-\alpha)+\alpha) \cdot S \cdot f' \cdot f$			

Table 6.6: Computation and memory demand for S minibatch, kxk kernel with f 'depth, nxn input feature with f depth.

and minibatch [Figure 6.33]. The advantage of the proposed method increases as the input image gets larger. Also, its complexity remains almost same regardless of the kernel size since the kernels are zero-padded to be the same size as the feature maps. Therefore, the proposed method motivates the use of large images and kernels in the network. Also note that the proposed approach demands the same memory space as the spatial approach. By eliminating the need for the Fourier transforms, it does not require additional memory for storing the frequency-domain parameters for their reuse.

Recognition Accuracy

The proposed approach approximates the spatial-domain CNN model in two ways. First, I use the spectral pooling and activation functions, which are approximated using linearity of FFT. Also, the feature map dimension reduces differently from the spatial domain CNN, as discussed in section 3.1. In order to evaluate the effect of these approximations, I compare the recognition accuracy of the LeNet-5 network with MNIST [112] and CIFAR-10 [131] datasets. As Figure 6.34 shows, the proposed approach achieves similar accuracy trend over training iterations as the original spatial-domain approach. While having the similar number of iterations, I reduce total training time and energy with less demand in computation and memory access per iteration.

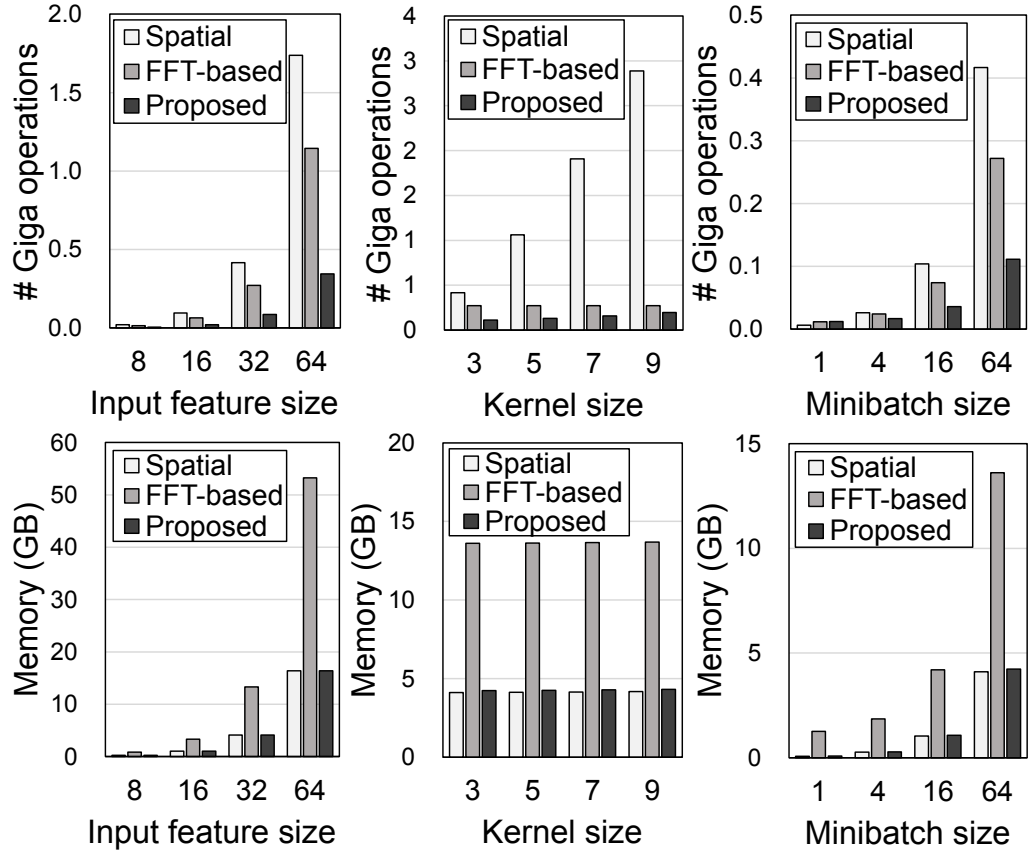


Figure 6.33: Computation and memory demand for a given convolution layer. Default values: input feature size=32x32 and depth=16, kernel size=3x3 and depth=16, and mini-batch=64.

6.2.5 Hardware Accelerator Design

Hardware Implementation

For training energy and latency analysis, I implement the proposed accelerator in a hardware [Figure 6.35(a)]. The 2-D FFT/IFFT module is created based on an array of 2-point 1-D FFT/IFFT implementation presented in [132]. For parallel multiplications of complex numbers, I compose an array of 72 complex-number multipliers. As Figure 6.35(b) shows, the computation engine is designed to accumulate each output feature plane after element-wise multiplication of an input feature and a kernel.

For comparison with the existing approaches, I also design a spatial-domain convolu-

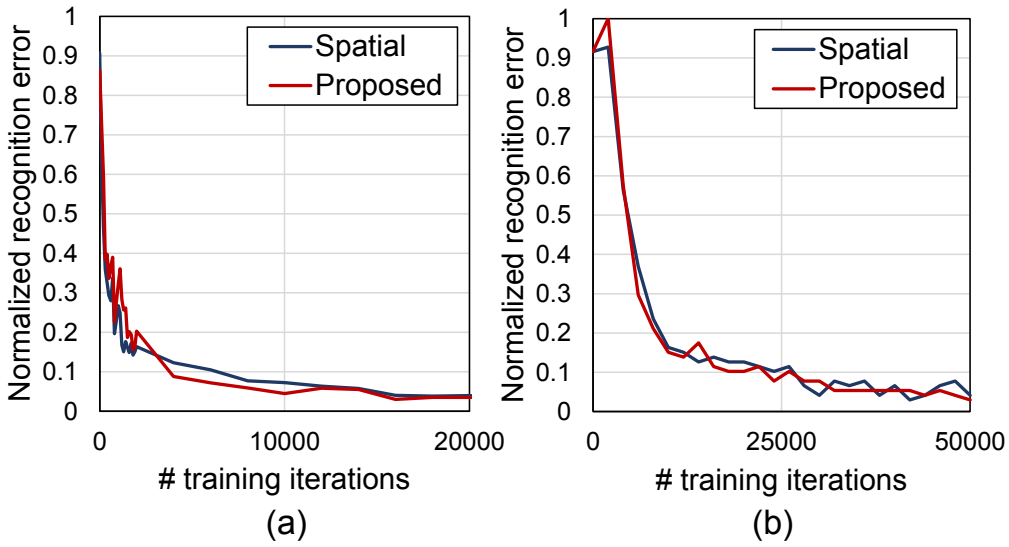


Figure 6.34: Normalized recognition error vs. number of iteration for (a) MNIST and (b) CIFAR-10 dataset.

tion module using a systolic array of MAC (multiply-and-accumulator) units as presented in [111]. The array includes 120 real-number MAC units, which occupies the same area (1mm2) as the array of 72 complex multipliers. The design is synthesized with 28nm CMOS technology for energy and latency analysis [Figure 6.36(a)], as well as in an FPGA to validate the FPGA compatibility [Figure 6.36(b)].

Energy and Latency Analysis

Table 6.7 shows the hardware cost of the three training approaches when applied to train the LeNet-5 and AlexNet networks for one-epoch with minibatch of 128. Compared to the spatial-domain training, the proposed approach significantly reduces the computational complexity while maintaining the same memory requirement. Figure 6.37 illustrates how I reduce the hardware demand by applying various techniques.

This reduction translates into $2\times$ lower latency and $2.5\times$ lower energy for LeNet-5 training [Figure 6.38]. Although the FFT-based convolution approach also reduces computation overhead of convolution, it requires a number of Fourier transforms that demand large amount of latency and energy. Furthermore, its large memory requirement makes it

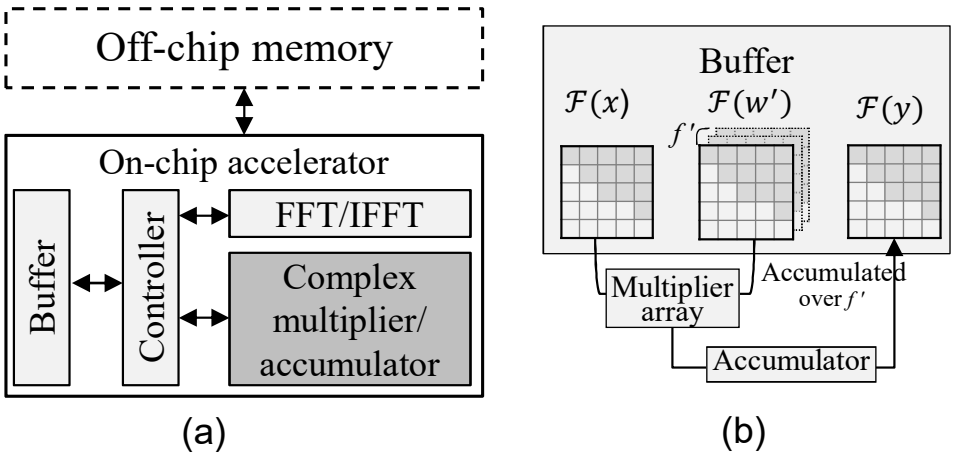
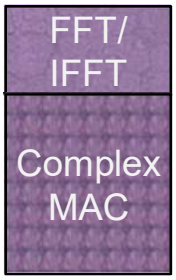


Figure 6.35: (a) Proposed accelerator design. (b) Multiplier and accumulator process. Bit width: 32-bit real/imaginary. Off-chip memory: DDR3 (70 pJ/bit access E, 12 ns/64 bit latency).

consume more time and energy to access the data from off-chip memory. As Figure 6.39 shows, the proposed approach achieves more reduction ($4.7\times$ latency and $6.3\times$ energy) in case of AlexNet, which has larger input image and kernels. Note that the proposed approach can be used to accelerate inference process as well. For AlexNet, the proposed accelerator requires $4.5\times$ and $6.0\times$ lower latency and energy than the spatial-domain convolution hardware.

6.2.6 Summary of the Section

Computational complexity is a major challenge in training CNNs using mobile platforms with limited resources. In response, this section designed an accelerator that trains the CNN using efficient frequency-domain computation. By mapping all the parameters and operations into the frequency domain, convolution is performed using simple pointwise



(a)

H/W	Resources	FFT/ IFFT	Complex MAC	Total
ASIC	Area (mm^2)	0.49	1	1.49
	Power (mW)	148	311	459
FPGA	Flip-flops	8,192	23,040	31,232
	Multipliers	176	360	536
	Adders	48	216	264
	Power (mW)	219.2	921.6	1140.8

(b)

Figure 6.36: (a) ASIC layout and (b) ASIC/FPGA area, power, and utilization of the proposed accelerator. (Operating frequency of ASIC: 500MHz, FPGA: 100 MHz).

multiplications without the Fourier transforms. I maintain the memory requirement same as the conventional approach by exploiting the FFT properties such as sinc interpolation and conjugate symmetry. With the reduced computational and memory overhead, it significantly reduces time and energy for both training and inference. The future work needs to consider integration of the precision control of the complex-number computation, as well as GPU-based implementation of the approach.

Network	Approach	Computation(GOPs)		Memory (MB)
		FFT/ IFFT	Conv (Mult)	
LeNet-5	Spatial	-	321.8	1.6
	FFT-based	33.7	113.1	9.1
	Proposed	**4.0**	**45.0**	**1.6**
AlexNet	Spatial	-	1,028,916	534
	FFT-based	28,223	317,316	4,594
	Proposed	**1,621**	**128,030**	**534**

Table 6.7: Computation/memory demand for one-epoch (128 minibatch) training of LeNet-5 and AlexNet.

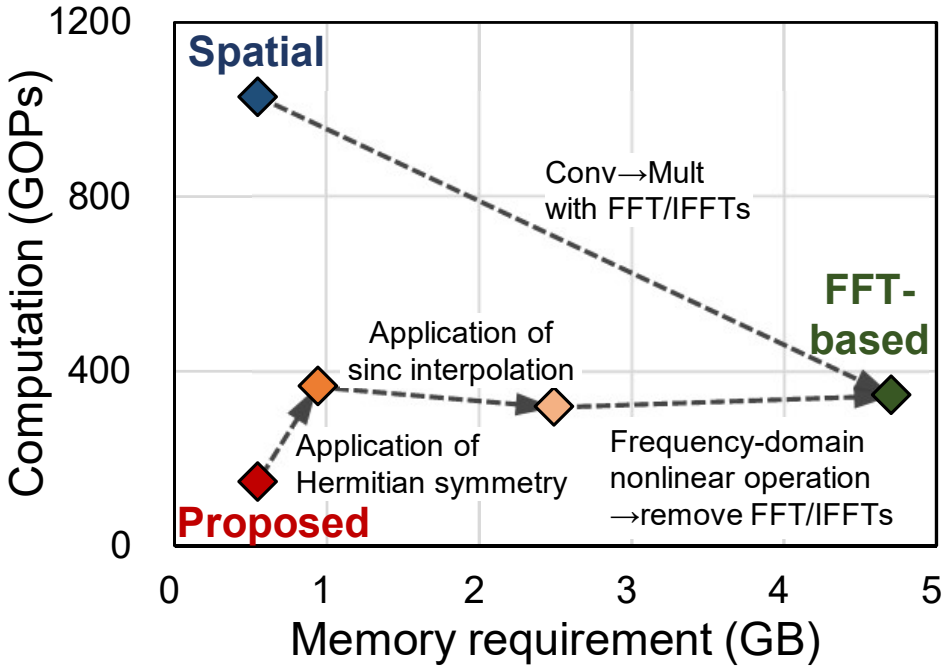


Figure 6.37: Computation/memory reduction of AlexNet training through the proposed techniques in this work.

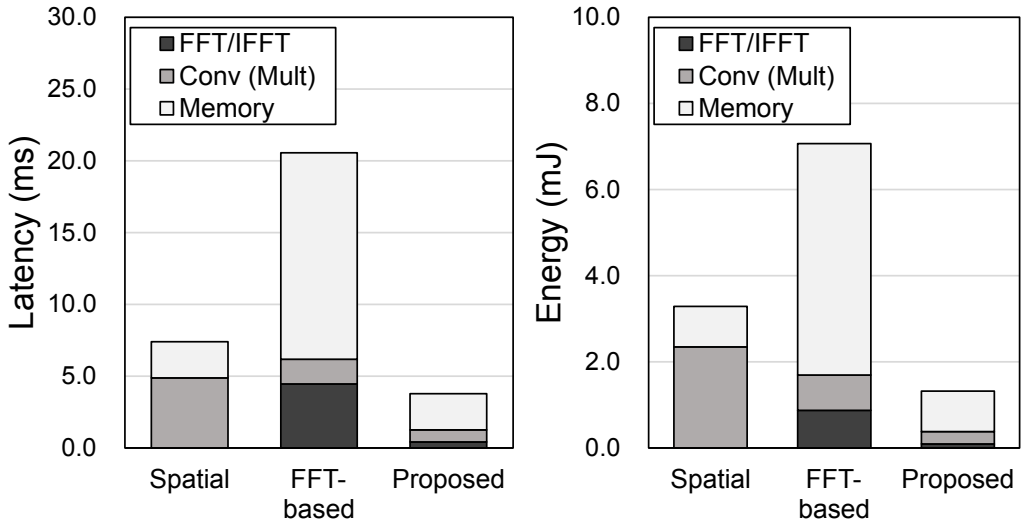


Figure 6.38: Latency and energy required for one-epoch (128 minibatch) training of LeNet-5.

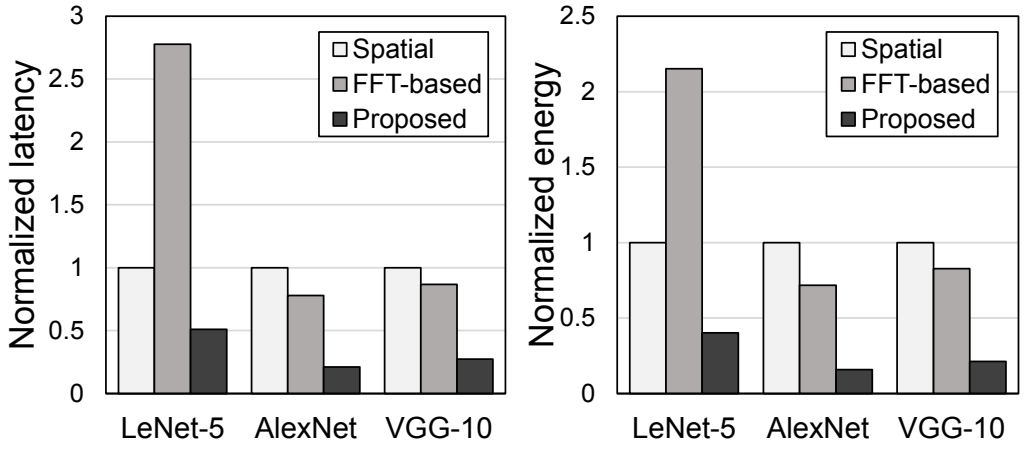


Figure 6.39: Latency and energy normalized to the spatial-domain approach.

CHAPTER 7

EDGE-HOST PARTITIONING OF DEEP NEURAL NETWORK INFERENCE

7.1 Introduction

The previous chapter assumed that the edge platforms run deep learning inference independently, so that they can provide a local feedback or decision directly at the edge, such as controlling sensor parameters. More general application model is that the edge platforms are connected to the host platform, where users make decisions based on the information processed through a neural network (e.g., video surveillance [60], remote monitoring [61]). This chapter focuses on this edge-host combined environment, aiming to make the information from neural network inference available at the host, using limited resources at the edge devices. At the same time, the information should be delivered with as high throughput as possible under a bandwidth constraint, to enable better decision at the host.

One configuration to achieve this goal is to utilize the edge platform as an image source that provides the visual data to the host so that it can perform neural network inference with relatively sufficient resources [Figure 1(a)]. The challenge is to achieve high image throughput with a limited transmission bandwidth. Therefore, the input image space is normally compressed (e.g. Motion JPEG) to reduce the bandwidth demand and transmission energy dissipation at the edge. To avoid bandwidth-intensive transmission between the edge and the host, an edge platform can embed a neural network inference engine to process the image data directly on device [Figure 1(b)]. By transmitting the end output of inference, the transmission demand can be significantly reduced. However, neural network inference is a costly operation that requires large memory and computation, which degrade the energy-efficiency and throughput of the edge platforms. Moreover, solving complex problems requires deeper networks and a large number of parameters, significantly increas-

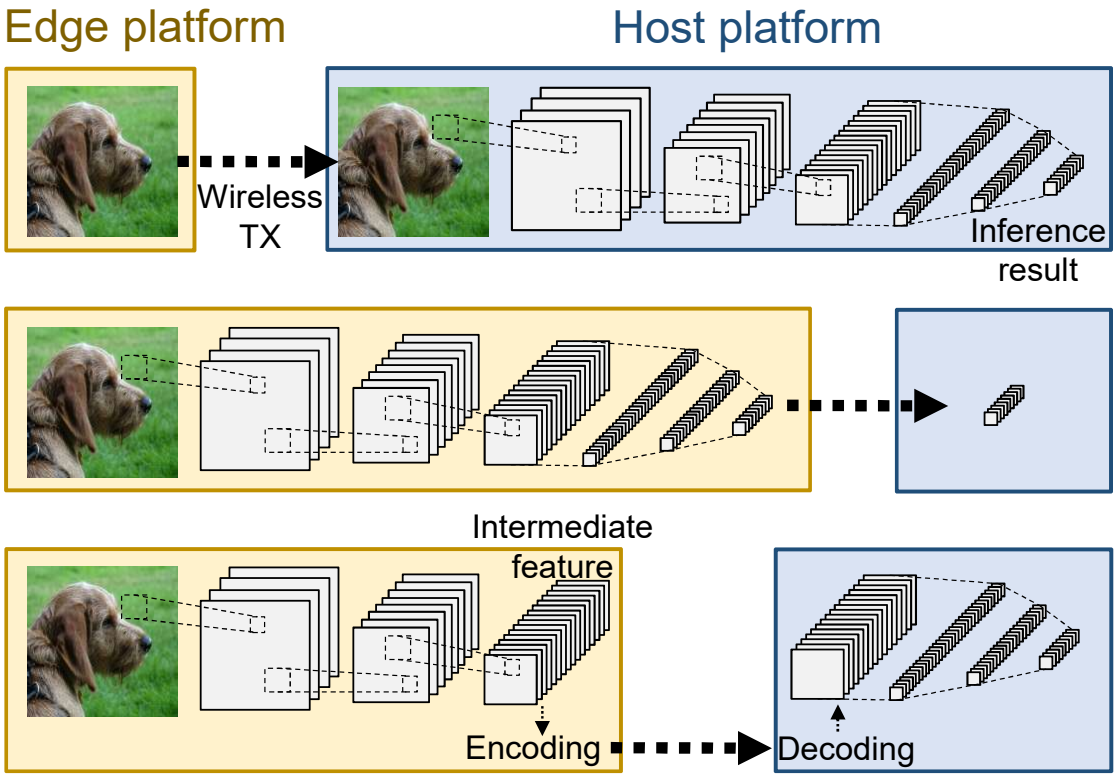


Figure 7.1: Three approaches to deep-learning inference in IoT environment with an edge and a host platform. (top) Entire inference at the host using images transmitted by the edge, (middle) entire inference at the edge that transmits the end output to the host, and (bottom) the proposed partitioned inference where the edge processes inference up to an intermediate layer, whose features are encoded and transmitted to the host for inference of the rest of the network.

ing the compute/memory demand at the edge device.

This section proposes partitioning a DNN between the edge and the host platform to perform coordinated inference to enhance the energy-efficiency (Frames/J) and throughput (Frames/second) of the edge under the channel bandwidth and accuracy constraints [Figure 7.1(c)]. I present a DNN as an information encoding pipeline that can be partitioned between the edge and the host at an intermediate layer. The output feature map of that intermediate layer is further encoded/compressed to reduce the data volume, and transmitted to the host. I show that from the perspective of the edge device, there is a clear trade-off

between the inference and transmission demand depending on how the inference task of a network is allocated. The major contributions of this section are:

- I introduce a partitioned inference approach with feature space encoding, where the edge platform processes inference up to an intermediate layer of the network, and transmit the output features to the host platform for inference of the rest of the network.

- Based on the tradeoff analysis of convolutional neural networks (CNNs), I propose a design guideline for partitioning that allocates convolutional layers at the edge and the rest fully-connected layers at the host.

- I introduce feature space encoding where the output features are compressed (lossless or lossy) before transmission to further enhance the bandwidth utilization. I characterize the accuracy impact of feature space encoding at different layers.

- I propose split re-training of the DNN to enhance the accuracy with feature space encoding. The partition at the host is fine-tuned by augmenting the training data with encoded features of the intermediate layer, while the partition at the edge remains unchanged.

The advantage of the proposed partitioning approach with feature encoding is demonstrated by the energy/throughput analysis based on an inference engine with an integrated JPEG encoder. The simulation with AlexNet shows that the JPEG-based lossy feature encoding method reduces the size of the last convolutional layer features of AlexNet by 28x with 1% accuracy loss. Fine tuning of the partitioned network further reduces the transmission demand by 11% at the same accuracy. I show that the proposed approach improves the system energy efficiency and throughput by 15.3x and 16.5x compared to performing entire inference at the host, and 2.3x and 2.5x compared to performing the entire inference at the edge. It is shown that the improvement depends on the network complexity as well

as the transmission bandwidth. Moreover, I show that, although partitioning at the last convolution layer is an efficient design approach, there is a need for dynamic control of the partitioning position to further enhance the throughput or energy.

In this section, I build on the prior studies by proposing a design guideline for network partitioning based on the layer-wise trade-off study on the energy-efficiency and throughput of the edge platform under accuracy and bandwidth constraints. The analysis of inference partitioning incorporates the effect of feature encoding in different layers of the network on the transmission demand and the classification accuracy.

7.2 Partitioning of Inference with Feature Space Encoding

Our system model includes an edge and a host platform, with the edge platform capturing visual data, which is processed through neural network inference, and the output is gathered at the host platform. Here, the goal is to make the neural network inference results available at the host platform at a target accuracy, with maximum energy efficiency and throughput at the edge device. To achieve this goal I propose partitioning a network at an intermediate layer, whose features are encoded and transmitted to the host for the rest of the inference.

7.2.1 DNN as an Information Encoding Pipeline

I view a DNN as an information encoding pipeline. Each layer in a DNN encodes the input features into the output feature space. The size of the output feature space reduces when it passes through a max-pooling operation. Moreover, as each layer encodes the information into a different feature space, the characteristics of the output feature maps such as sparsity or entropy change as well [Figure 7.2]. Therefore, if conventional signal/image encoding and compression techniques are applied at the output feature maps, we can expect different responses at different layers.

With the preceding view of a DNN as an information encoding pipeline, the edge platform can now be regarded as an encoding engine that processes the raw visual data into a

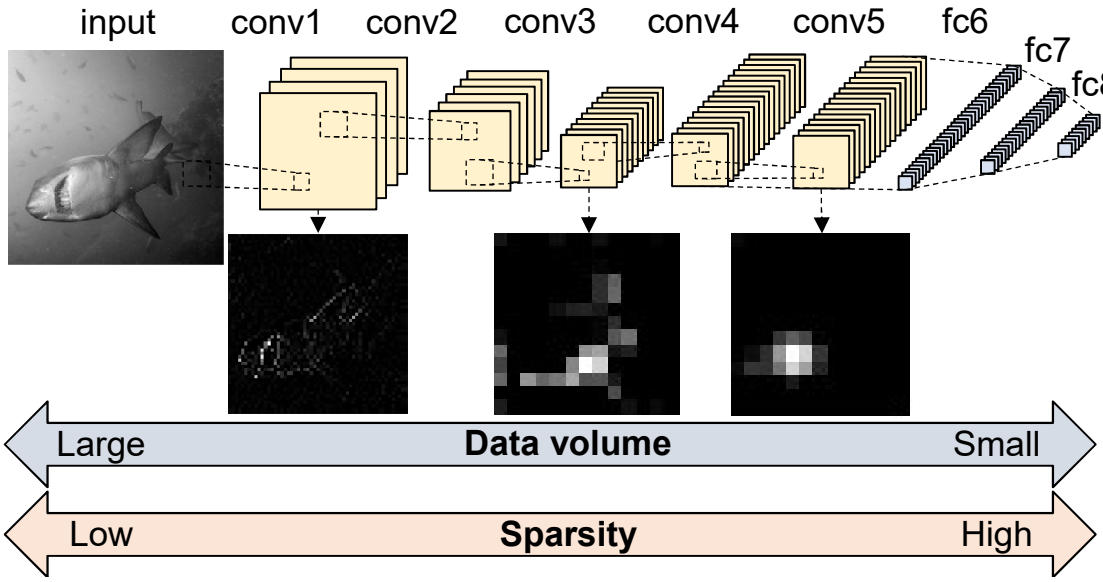


Figure 7.2: Visualization of one of the feature maps in different layers of AlexNet, indicating the sparsity increases as the feature is in deeper layers.

different space before delivering it to the host. The processing involves information extraction through DNN layers and data compression through algorithmic encoding (e.g., JPEG) of the features. The overall compression level is controlled by the parameters, namely, how many layers are processed at the edge (partitioning position) and quality factor (QF) of the algorithmic encoder.

7.2.2 Edge-Host Partitioning of Inference

I first demonstrate the tradeoff analysis of the partitioning approach based on the inference/transmission demand of AlexNet when it is partitioned at different layers [Figure 7.3]. When the entire inference is performed at the host platform, the edge platform should transmit the input images to the host. Figure 7.3(b) shows that the input feature size for AlexNet is nearly 0.3 MB with 16-bit precision. To satisfy a typical frame rate of 30 frames/sec, the required bandwidth can go up to 70 Mbps, which is not generally accommodated by conventional low-power transmitters. To mitigate the transmission overhead, the edge platform can process the whole network inference and transmit the output of the network. However,

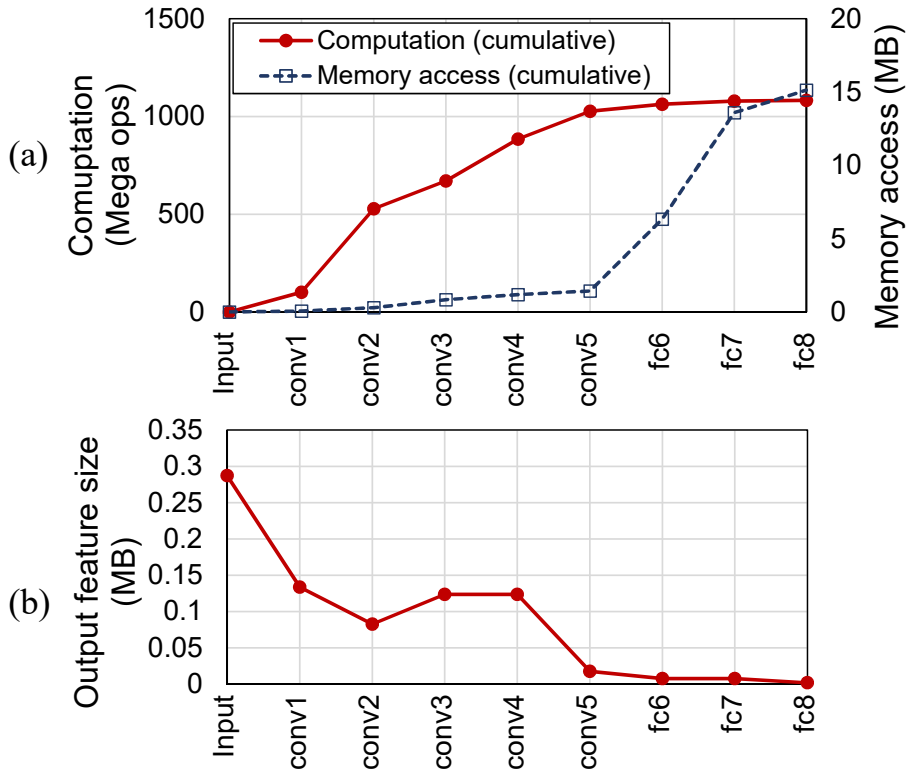


Figure 7.3: (a) Cumulative computation and memory access (for compressed weights) demand of inference and (b) output feature size for each layer of AlexNet.

the entire network inference requires huge computation and memory demand, as illustrated in Figure 7.3(a).

By allowing transmission of the intermediate features, the inference partitioning approach enables the layer-wise tradeoff for higher energy-efficiency and throughput at the edge. The system performance with inference partitioning will be determined depending on which layer the network is partitioned. If we partition the network at a deeper layer of a CNN by including fully-connected layers at the edge, the system throughput and energy efficiency will degrade mainly because of the memory access demand of fully connected layers. If we partition the network at an earlier convolutional layer, transmission demand will be still limiting the system performance because of its larger feature size.

Based on these observations, I propose an initial design guideline that partitions the network at the end of convolution layers. An edge device designed to perform convolutional

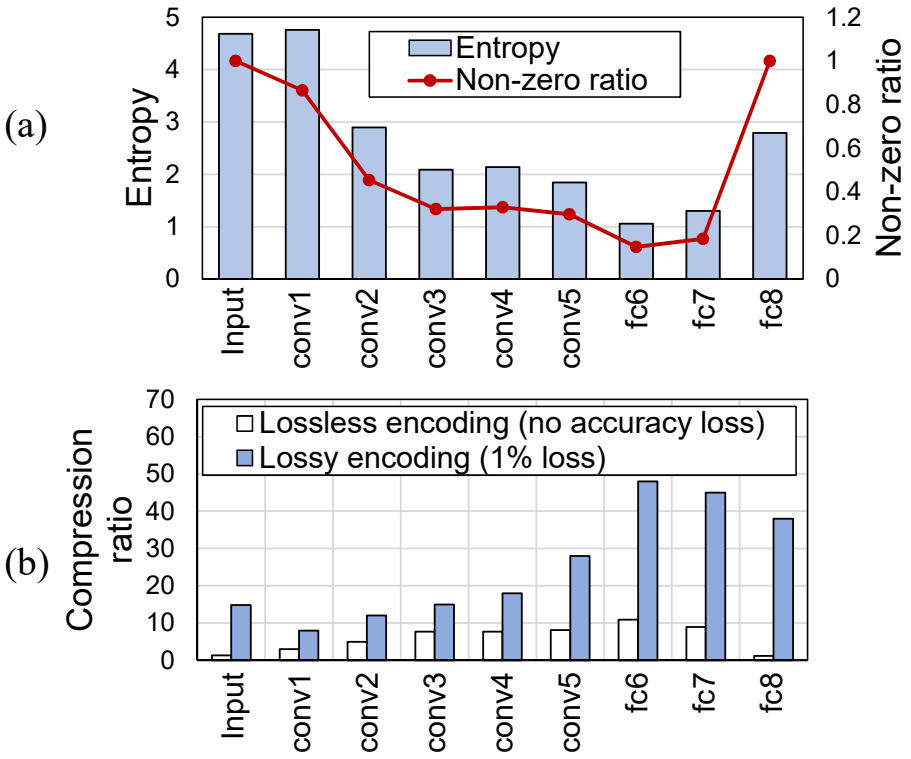


Figure 7.4: (a) Entropy and non-zero ratio of features in different layers in AlexNet. (b) Compression ratio of each layer features with lossless and lossy encoding.

layer inference will be useful because a set of convolutional layers is widely used as a feature extractor for backbone networks such as Convolutional-RNN. Another benefit of this partitioning approach is that the data flow of the edge platform can be optimized for convolutional layers instead of considering the heterogeneous data flow in fully-connected layers. Also, smaller kernel size of the convolutional layers makes them easily fit into the edge devices with limited storage.

7.2.3 Feature Space Encoding

Inference partitioning at an intermediate layer implies that its output feature maps have to be transmitted to the host side. To further improve energy-efficiency and throughput, I propose encoding the feature maps before transmission.

A primary reason for encoding the intermediate features is to reduce the transmission

demand. Moreover, data representation of the original feature maps is generally sparse, making them easily compressed. Figure 7.4(a) shows that the sparsity increases as the features are in deeper layers, with lower non-zero ratio and entropy. However, the feature encoding may cause loss of information, which leads to the degradation of the inference accuracy. Therefore, to leverage the benefit of feature encoding while minimizing the loss of accuracy, I examine the robustness of each layer features to different encoding methods including lossless and lossy encoding.

Lossless Encoding

Sparse representation of the feature maps is appealing property for lossless encoding. I apply a common lossless encoding method, run-length encoding combined with the Huffman encoding, which is the last part of the JPEG encoder pipeline. As this method converts sequences or zeros into the pre-defined codes, higher ratio of zeros in deeper layer features leads to higher compression ratio, as illustrated in Figure 7.4(b). Although the encoded features are perfectly reconstructed at the host, resulting in no accuracy loss, its compression ratio is limited to 3-10x.

Lossy Encoding

I apply JPEG encoding (lossy) to the intermediate feature, to achieve further compression of the features at the expense of the loss of accuracy. Figure 7.4(b) show the compression ratio of each layer features obtained at the 1% accuracy loss. As lower entropy leads to higher performance of lossy encoding, deeper layer features with lower entropy generally shows higher compression ratio. With 1% accuracy loss, lossy encoding achieves 5-50x compression, which significantly reduces the demand on transmission bandwidth and energy.

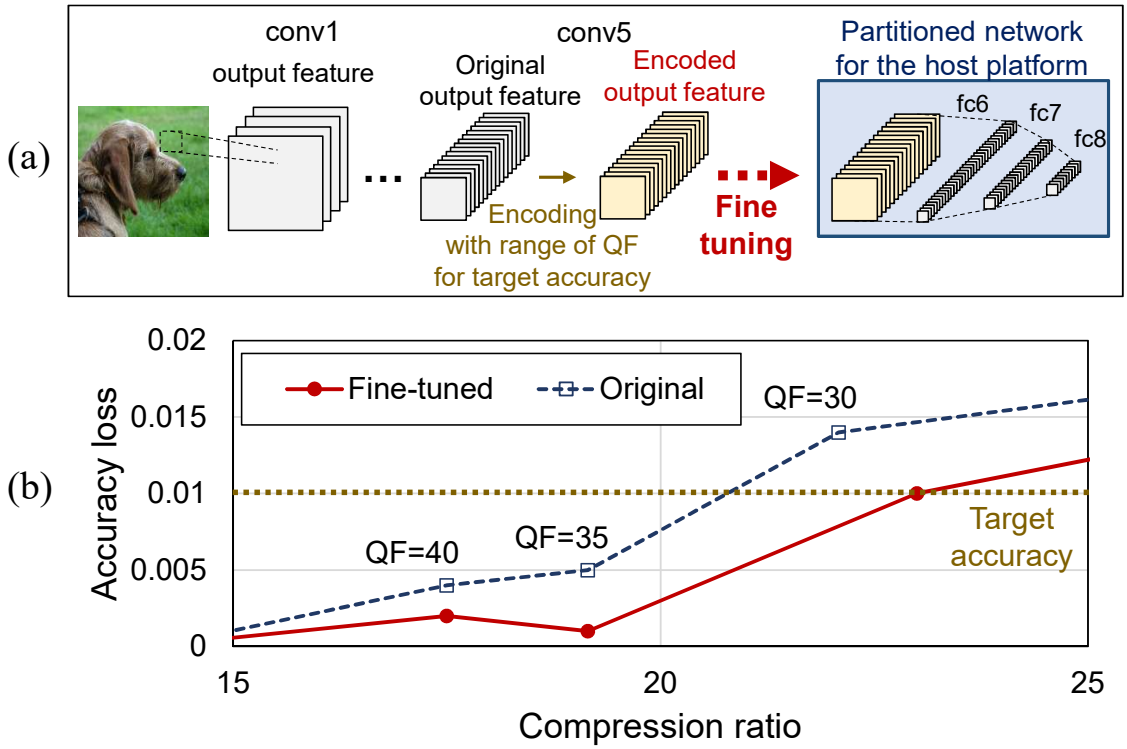
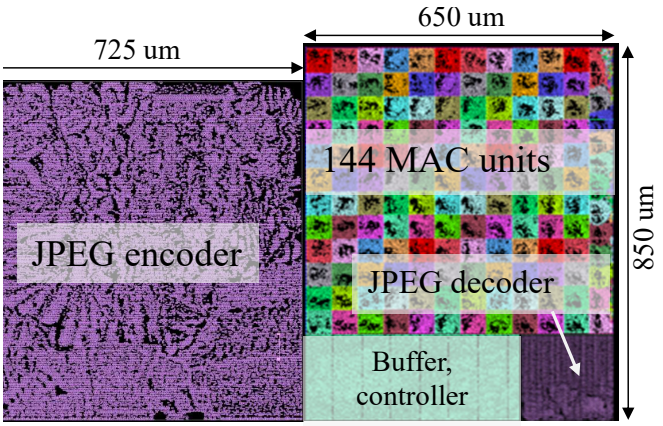


Figure 7.5: (a) Process of fine tuning with encoded intermediate features. (b) Compression ratio and accuracy loss of a partitioned network according to various QF values used to encode the conv5 output features.

Fine-Tuning of the Partitioned Network

I propose to apply a fine-tuning technique to enhance the accuracy under lossy encoding of the features. In other words, improved accuracy due to fine-tuning will allow higher compression for the same accuracy loss. Fine tuning is commonly used technique to enhance the accuracy of the entire network using augmentation of input space. However, since the transformation is applied at an intermediate layer, I propose split re-training of the network, where *only the network partition at host is re-trained*, while the edge partition remains same [Figure 7.5(a)]. The augmentation of the input to the partitioned network is determined by the compression performance of the original features of the target layer. Figure 7.5(b) shows an example of the AlexNet conv5 layer, where the target accuracy (1% loss) with the original network is achieved with the quality factor (QF) between 30 and 35.



(a)

Module	Area (mm^2)	Power (mW)
JPEG encoder	0.526	3.5
JPEG decoder	0.032	0.88
MAC units	0.436	4.32
Buffer/controller	0.085	0.48
Total	**1.079**	**9.18**

(b)

Figure 7.6: Hardware design of an inference engine. (a) Layout of the synthesized design and (b) area and power breakdown of the system.

Therefore, I prepare the input features for fine tuning by encoding the intermediate features of the training dataset with the randomly selected QF between 30 and 35. The encoded features are then supplied to fine-tune the network partition for the host platform. As Figure 7.5(b) shows, fine tuning further enhances the compression ratio by 11% at 1% accuracy loss.

7.3 Simulation Results

7.3.1 Inference Engine Design and Modeling

I designed an inference engine that performs inference and feature encoding to enable the energy and throughput analysis of the proposed approach. The engine includes an array of 144 16-bit MAC units, a JPEG encoder for feature space encoding, and an on-chip buffer to store the input/output feature maps [Figure 7.6(a)]. As I assume the weights are stored in a compressed format through the JPEG encoding method presented in [133], I included a JPEG decoder for decoding the compressed weights. The design was synthesized into an ASIC with a 28nm process, and its area and power characteristics are shown in Figure 7.6(b)

For the weight storage, I use an off-chip LPDDR DRAM with 640 pJ/32 bit data access

and 12.8 MB/s bandwidth [134] [135]. Feature transmission is performed by an ultra low power 2.4GHz 802.11bgn transmitter module [136] that has 1, 2, and 22 Mbps datarate modes with 62.7, 99, and 660 mW power consumption, respectively (I use a 2 Mbps mode as a default). As I model the processing engine of 2-stage pipeline with inference and feature encoding/transmission stages, the system throughput is determined by the largest latency among these two stages.

7.3.2 Energy/Throughput Analysis

Analysis without Feature Space Encoding

Figure 7.7(a) and 7.8(a) show the system energy/throughput when AlexNet is partitioned at different layers. For inference at the host (i.e., input feature transmission from edge), I assume the edge applies JPEG encoding on the raw image (input feature). The quality factor for the input compression is estimated by considering 1% accuracy loss at the classification. Even with 15x compression of the input features, the energy/throughput performance of the host inference approach is still limited by transmission. Next, I consider processing a set of CNN layers at the edge. Here, I assume no feature encoding at intermediate layers. When the network is partitioned at a deeper layer, inference becomes the system bottleneck.

By including only a part of the network at the edge, the inference partitioning approach enhances the system performance. The maximum throughput and minimum energy of the edge platform is realized when the network is partitioned after the first fully-connected layer (fc6). At this point, the energy-efficiency and throughput is enhanced by 1.2x and 1.15x compared to the entire inference at the edge, and 4.5x and 7.5x compared to the entire inference at the host, respectively.

Analysis with Feature Space Encoding

When the network is partitioned at convolutional layers, their large feature sizes still make the transmission become the throughput and energy bottleneck. Figure 7.7(b) and 7.8(b)

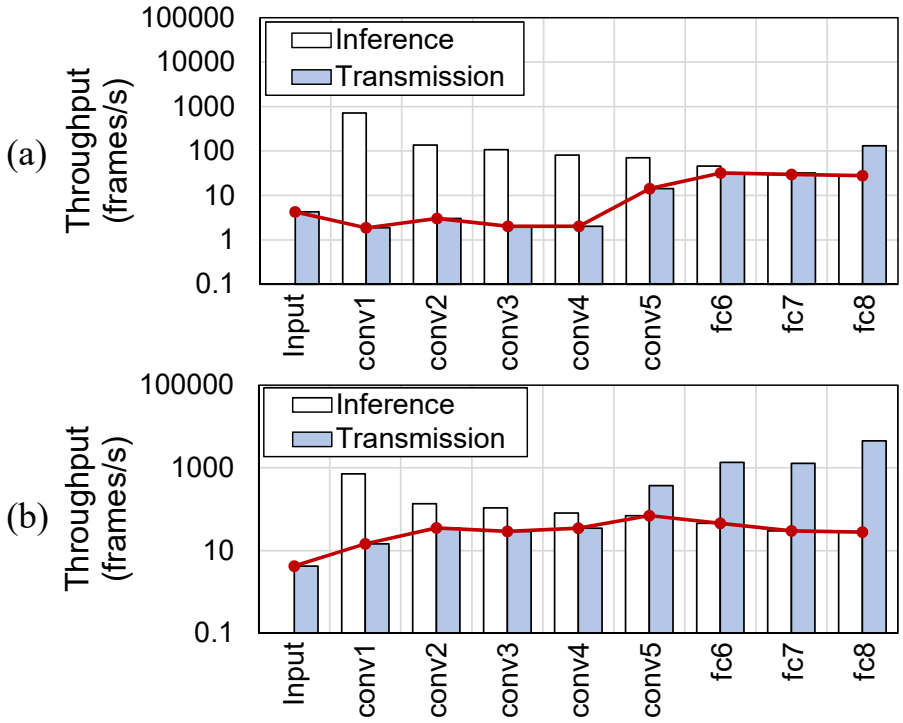


Figure 7.7: System throughput of the edge platform. (a) Without encoding and (b) with lossy feature encoding (at the accuracy loss=1%) and weight compression. Here, the encoding latency is included in the transmission latency.

show that the feature encoding approach reduces transmission latency and energy consumption with negligible encoding overhead. The improved transmission throughput enhances the system throughput in convolutional layers where transmission was the bottleneck in the previous case without feature encoding. Feature encoding significantly enhances the system energy as well when the network is partitioned at convolutional layers.

With feature encoding, the optimal partitioning layer for maximum throughput and minimum energy becomes the last convolutional layer (conv5). If the network is partitioned at fully-connected layers, the benefit of feature encoding is limited because their original feature sizes are relatively small and the system energy is dominated by inference. If we partition the network at an earlier layer, transmission is still the bottleneck because of their larger feature size. In these layers, the transmission throughput can be improved by compressing the features further, but further compression will result in more accuracy loss.

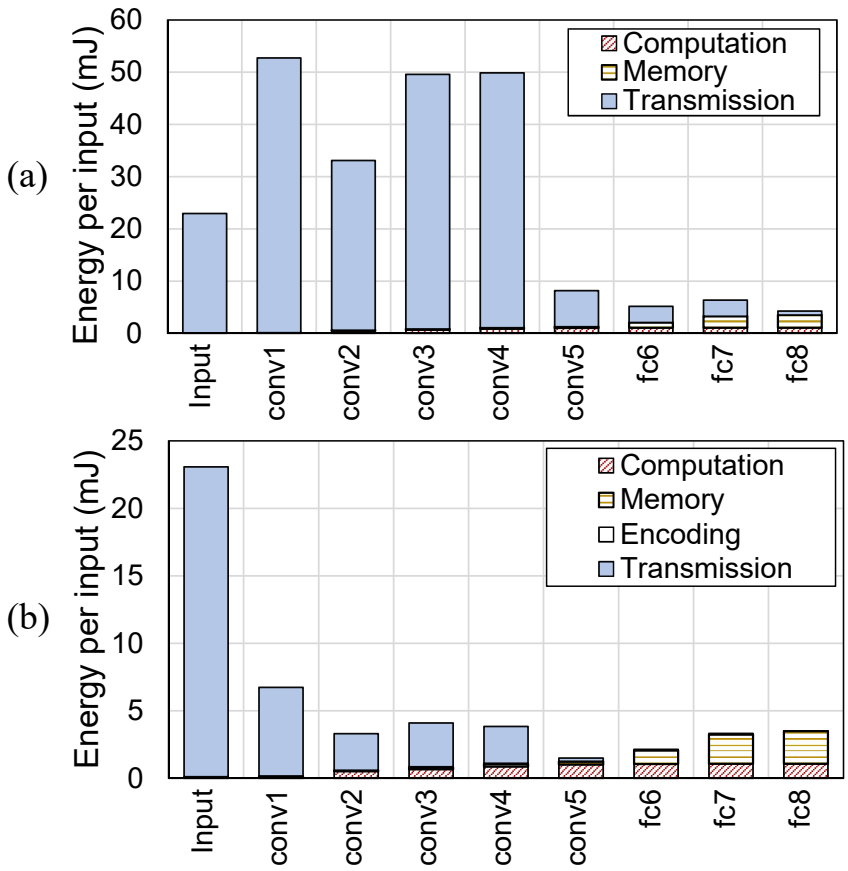


Figure 7.8: System energy breakdown of the edge platform. (a) Without encoding and (b) with lossy feature encoding (at the accuracy loss=1%) and weight compression.

Figure 7.9 illustrates the summary of performance improvement obtained by the proposed partitioned inference with feature encoding on AlexNet. Higher compression of the features through lossy encoding combined with fine tuning provides more gain in energy efficiency and throughput. The proposed partitioning approach with fine-tuning assisted feature encoding achieves energy reduction and throughput improvement by 15.3x and 16.5x than the entire inference at the host, and 2.3x and 2.5x than the entire inference at the edge, respectively.

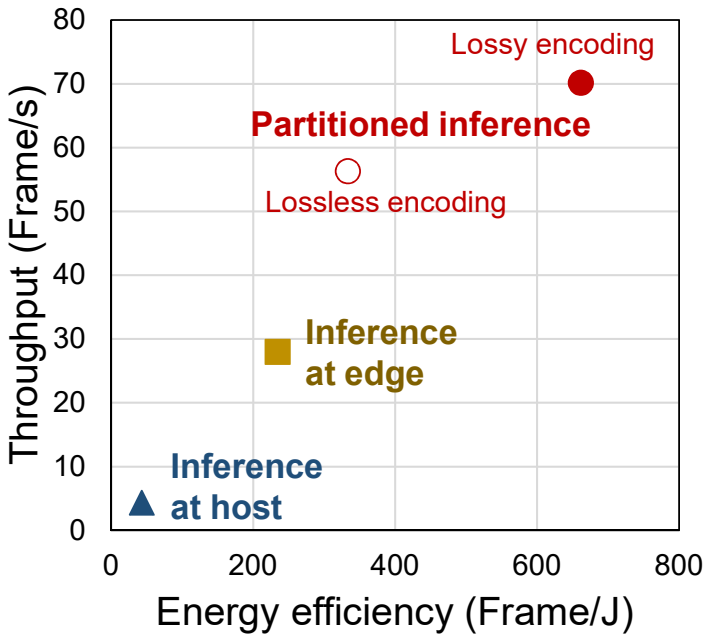


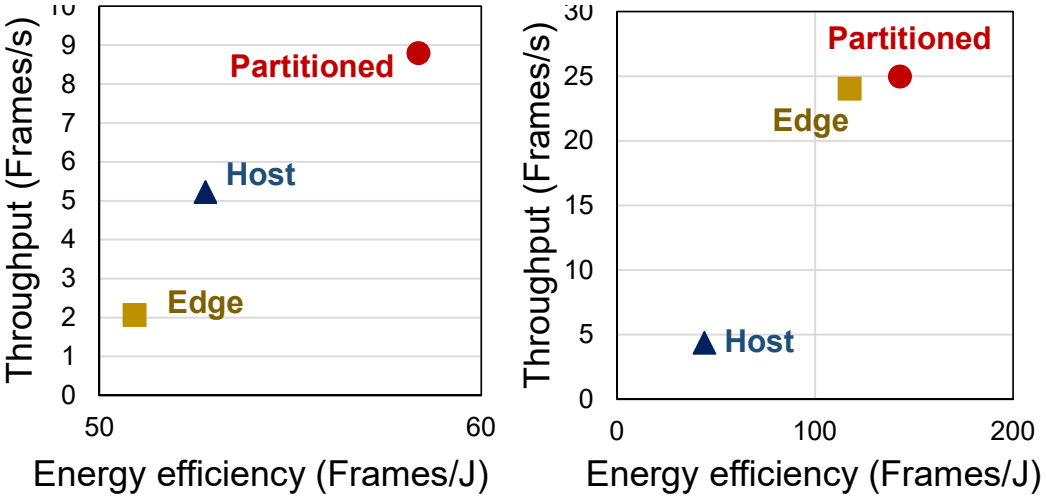
Figure 7.9: Summary of improvement of the throughput and energy-efficiency of the proposed partitioning with feature encoding.

7.3.3 Discussions

Effect of Network Types

The peformance of the proposed inference partitioning approach will vary depending on the type of the neural network. I perform the simulation with two other network models, VGG-16 [137] and ResNet-50 [138] [Figure 7.10]. In VGG-16, large memory access demand of fully-connected layers significantly degrades the performance of the edge inference approach, compared to the AlexNet case. By including only convolutional layers at the edge, the proposed partitioning approach can avoid large memory access of fully-connected layers, achieving 1.2x and 4.3x higher energy efficiency and throughput than the edge inference model.

ResNet-50 has deep convolutional layers followed by one fully-connected layer, with smaller feature sizes making its computation demand smaller than VGG-16. Therefore, instead of inference at the host, processing the entire inference at the edge leads to higher



(a) (b)

	Computation (Giga ops)		Memory (MB)	
	Conv	FC	Conv	FC
AlexNet	1.0	0.06	1.43	13.7
VGG-16	14.6	0.12	8.2	22.2
ResNet-50	3.5	0.002	22.1	0.7

(c)

Figure 7.10: Throughput energy efficiency of each approach for (a) VGG-16 and (b) ResNet-50. (c) Inference demand of each network (assuming compressed weights).

throughput and energy efficiency. The proposed partitioning approach provides better efficiency by removing the last layer inference from the edge, since it can avoid large memory demand of the fully-connected layer.

Effect of Transmission Channel

Figure 7.11 shows the performance of each approach with two different transmission channels (low bandwidth=1 Mbps, high bandwidth=22 Mbps) on AlexNet. Even with a transmission bandwidth increase, the performances of the edge inference and the partitioned inference approaches remain almost same because they are not transmission-bottlenecked. On the other hand, host inference approach enhances the system throughput with reduced transmission latency of the input images. We observe that the convolutional layer partitioning approach provides performance gain over the edge or host inference approaches.

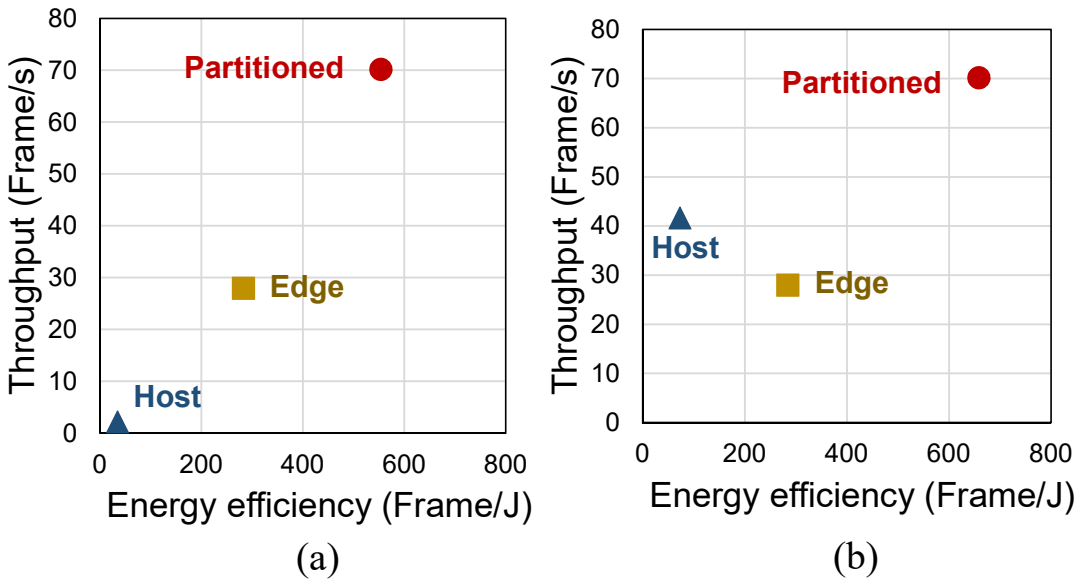


Figure 7.11: Throughput energy efficiency of each approach for (a) Low BW (1 Mbps) and (b) high BW (22 Mbps) channel.

I further characterize the effect of transmission channel on the network partitioning. Figure 7.12 shows how the optimal partitioning layer is shifted according to the transmission bandwidth and energy consumption of a transmitter. We observe that in a large-bandwidth case, the optimal partitioning layer for maximum throughput is shifted to the second layer since transmission can accommodate larger features. However, with a lower channel bandwidth, it is preferred to process more layers by partitioning at deeper layers. Although convolutional layer partitioning continues to provide effective gain in a wide range of commercial low-power transmitter models, I believe the future work can include designing a controller that dynamically adapts to the variation of transmission channel by shifting the partitioning layer, together with controlling the data volume at a certain layer.

7.4 Summary of the Section

In this section, I introduced partitioning a DNN inference task between the edge and the host platforms to improve the energy-efficiency and throughput of the edge device. The improvement is leveraged by encoding the features of an intermediate layer, with the help

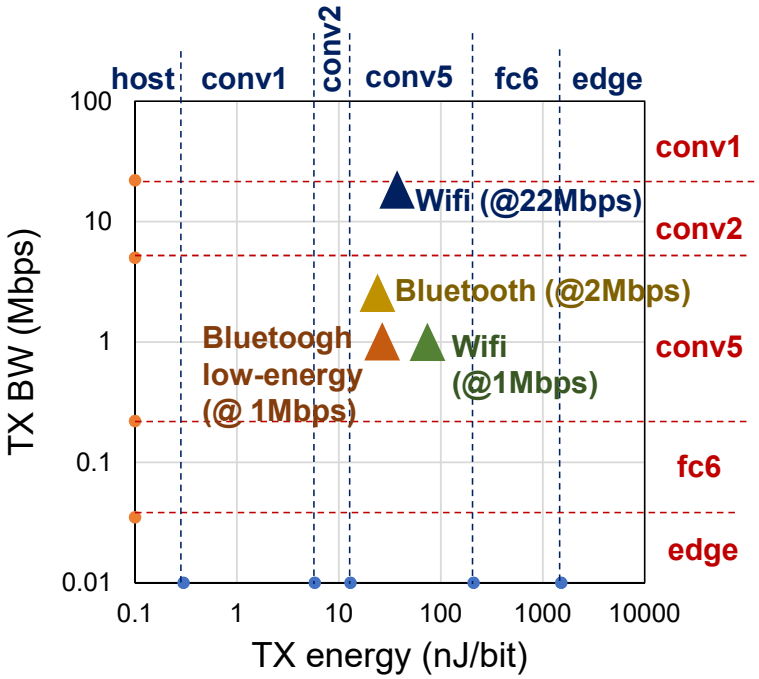


Figure 7.12: The optimal partitioning layer in terms of the throughput (y-axis) and energy-efficiency (x-axis) according to the transmission BW and energy of a transmitter.

of fine tuning of the partitioned network. This section also demonstrated that the proposed partitioning coupled with feature encoding significantly enhances the energy-efficiency and throughput of the edge compared to the entire inference processing at the edge of at the host platform.

CHAPTER 8
CONCLUSION

This chapter summarizes the key contributions of this dissertation, and also identifies related future research goals.

In this thesis, ROI-based processing and data rate control schemes are presented for resource-aware and robust image sensor system implementation. The thesis also presented novel techniques such as adaptive weight compression, frequency-domain convolution, and inference partitioning for resource-efficient deep learning inference at the IoT edge devices.

Chapter 3 presented an wireless image sensor node with low-overhead pre-processing that enables reliable and energy-efficient transmission of visual information. The proposed design achieves low-power consumption as well as tolerance to variation in environmental conditions such as a wireless channel. The chapter also presented a single-chip image sensor node that can harvest energy from the reconfigurable on-chip dual-purpose image sensor. To further improve the self-power performance, the system adopted low-power circuit techniques such as block-level pipelining, power gating, and adaptive supply control. The system demonstrates the feasibility of a self-powered sensor at a low frame rate. The future work will include improving the self-supported frame rate by analyzing the effects of wireless transmission, PMU efficiency, and image pixel size on the self-power performance. The future work will also involve techniques to reduce the sensors noise and enhance noise-robustness of the image-processing algorithm.

Chapter 4 presented a noise-robust moving object detection method based on simple operations that require low memory and computation. Improved noise robustness of the ROI detection method reduces transmission energy with a reliable delivery of ROI. The chapter also presented an energy-efficient parameter-controllable ROI-based coding scheme that enhances the ROI quality with lower system energy consumption. Its better rate-quality

characteristic for non-ROI provides higher ROI quality with low computation energy. The chapter analyzed that improved energy-efficiency of the proposed approach leads to better energy-accuracy scalability for the intelligent surveillance system using deep-learning based object classification. Future work will include exploring better ROI coding method suitable for deep-learning based classification, and utilizing a feedback from the neural network to further optimize ROI-based processing.

Chapter 5 tackled unreliable nature of wireless communication in the IoT edge devices by adapting wireless channel parameters. The proposed system-wide feedback control scheme maintains the energy consumption of the system under variations in channel conditions. Once the target data rate is determined, the PID-based rate controller minimizes buffer size by dynamically tuning the coding parameters for the target rate. With its improved energy-efficiency, the proposed approach can be an attractive solution for reliable delivery of visual information in resource-constrained video sensor nodes for surveillance applications.

Chapter 6 presented an energy efficient design of a neural network inference engine based on the adaptive weights compression using JPEG image encoding. By adaptively determining the quantization level of JPEG, the proposed approach achieves high compression ratio while preserving error-sensitive information. The high compression leads to the throughput improvement of the inference hardware system at significantly lower energy. The chapter also presented an accelerator that performs inference and training of CNNs using efficient frequency-domain computation. By mapping all the parameters and operations into the frequency domain, convolution is replaced with simple pointwise multiplications without the Fourier transforms. The memory requirement is remained same as the conventional approach by exploiting the FFT properties such as sinc interpolation and conjugate symmetry. With the reduced computational and memory overhead, it significantly reduces time and energy for both training and inference. The future work needs to consider integration of the precision control of the complex-number computation, as well

as GPU-based implementation of the approach.

Chapter 7 introduced partitioning a DNN inference task between the edge and the host platforms to improve the energy-efficiency and throughput of the edge device. The improvement is leveraged by encoding the features of an intermediate layer, with the help of fine tuning of the partitioned network. We demonstrated that the proposed partitioning coupled with feature encoding significantly enhances the energy-efficiency and throughput of the edge compared to the entire inference processing at the edge of at the host platform. The future work will include generalizing the partitioning approach into a distributed environment with multiple edge devices.

Appendices

APPENDIX A
PUBLICATION LIST

A.1 Journal publications

[J1] **Jong Hwan Ko**, Burhan A. Mudassar, and Saibal Mukhopadhyay, "An Energy-Efficient Wireless Video Sensor Node for Moving Object Surveillance," IEEE Transactions on Multi-Scale Computing Systems (TMSCS), vol. 1, no. 1, pp. 718, Sep. 2015.

[J2] **Jong Hwan Ko**, Mohammad F. Amir, Khondker Zakir Ahmed, Taesik Na, and Saibal Mukhopadhyay, "A Single-Chip Image Sensor Node with Energy Harvesting from a CMOS Pixel Array," IEEE Transactions on Circuits and Systems I (TCAS-I), Vol. 64, no. 9, pp. 2295-2307, June 2017.

[J3] Taesik Na, **Jong Hwan Ko**, and Saibal Mukhopadhyay, "Clock Data Compensation Aware Digital Circuits Design for Voltage Margin Reduction," IEEE Transactions on Circuits and Systems I (TCAS-I), Vol. 64, no. 9, pp. 2401-2413, June 2017.

[J4] **Jong Hwan Ko**, Duckhwan Kim, Taesik Na, and Saibal Mukhopadhyay, "Design and Analysis of a Neural Network Inference Engine based on Adaptive Weight Compression," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD), Feb. 2018.

[J5] Mohammad F. Amir, **Jong Hwan Ko**, Taesik Na, Duckhwan Kim, and Saibal Mukhopadhyay, "3D Stacked Image Sensor with Deep Neural Network Computation," IEEE Sensors Journal, Mar. 21, 2018.

[J6] Arvind Singh, Nikhil Chawla, **Jong Hwan Ko**, Monodeep Kar, and Saibal Mukhopadhyay, "Energy Efficient and Side-Channel Secure Cryptographic Hardware for IoT-edge Nodes," IEEE Internet-of-Things Journal (IoT-J), accepted for publication.

[J7] **Jong Hwan Ko**, Taesik Na, and Saibal Mukhopadhyay, "An Energy-Quality Scalable Wireless Image Sensor Node for Object-Based Video Surveillance," IEEE Journal on Emerging and Selected Topics in Circuits and Systems (JETCAS), May 2018.

A.2 Conference presentations

[C1] **Jong Hwan Ko**, Burhan Mudassar, and Saibal Mukhopadhyay, "Adaptive Wireless Video Sensor Node Using Content-Aware Pre-Processing for Moving Target Identification," 40th Annual Government Microcircuit Applications and Critical Technology Conference (GOMACTech 2015), Mar. 2015.

[C2] Arvind Singh, Monodeep Kar, **Jong Hwan Ko**, and Saibal Mukhopadhyay, "Exploring power attack protection of resource constrained encryption engines using integrated low-drop-out regulators," 2015 International Symposium on Low Power Electronics and Design (ISLPED 2015), Aug. 2015.

[C3] **Jong Hwan Ko**, Khondker Zakir Ahmed, Mohammad Faisal Amir, and Saibal Mukhopadhyay, "A Self-Powered Wireless Video Sensor Node for Moving Object Surveillance," 41th Annual Government Microcircuit Applications and Critical Technology Conference (GOMACTech 2016), Mar. 2016.

[C4] **Jong Hwan Ko** and Saibal Mukhopadhyay, "An Energy-Aware Approach to Noise-Robust Moving Object Detection for Low-Power Wireless Image Sensor Platforms," Inter-

national Symposium on Low Power Electronics and Design (ISLPED 2016), Aug. 2016, Best paper award (2/190).

[C5] **Jong Hwan Ko**, Taesik Na, and Saibal Mukhopadhyay, "An Energy-Efficient Wireless Video Sensor Node with a Region-of-Interest Based Multi-Parameter Rate Controller for Moving Object Surveillance," 2016 IEEE Advanced Video and Signal-based Surveillance (AVSS 2016), Aug. 2016.

[C6] Khondker Zakir Ahmed, Mohammad Faisal Amir, **Jong Hwan Ko**, and Saibal Mukhopadhyay, "Reconfigurable 96x128 Active Pixel Sensor with 2.1W/mm2 Power Generation and Regulated Multi-Domain Power Delivery for Self-Powered Imaging," 42th European Solid-State Circuit Conference (ESSCIRC 2016), Sep. 2016.

[C7] **Jong Hwan Ko**, Mohammad Faisal Amir, Taesik Na, and Saibal Mukhopadhyay, "A Low-Power Wireless Image Sensor Node with Noise-Robust Moving Object Detection and a Region-of-Interest Based Rate Controller," 42th Annual Government Microcircuit Applications and Critical Technology Conference (GOMACTech 2017), Mar. 2017.

[C8] Taesik Na, **Jong Hwan Ko**, and Saibal Mukhopadhyay, "Clock Data Compensation Aware Clock Tree Synthesis in Digital Circuits with Adaptive Clock Generation," Design, Automation, and Test in Europe (DATE 2017), Mar. 2017.

[C9] **Jong Hwan Ko**, Duckhwan Kim, Taesik Na, Jaeha Kung, and Saibal Mukhopadhyay, "Adaptive Weight Compression for Memory-Efficient Neural Networks," Design, Automation, and Test in Europe (DATE 2017), Mar. 2017.

[C10] **Jong Hwan Ko**, Burhan Mudassar, Taesik Na, and Saibal Mukhopadhyay, "De-

sign of an Energy-Efficient Accelerator for Training of Convolutional Neural Networks using Frequency-Domain Computation," Design Automation Conference (DAC 2017), June. 2017.

[C11] Taesik Na, **Jong Hwan Ko**, Jaeha Kung, and Saibal Mukhopadhyay, "On-Chip Training of Recurrent Neural Networks with Limited Numerical Precision," International Joint Conference on Neural Networks (IJCNN 2017), June. 2017.

[C12] (Invited) **Jong Hwan Ko**, Yun Long, Mohammad Faisal Amir, Duckhwan Kim, Jaeha Kung, Taesik Na, Amit Trivedi, and Saibal Mukhopadhyay, "Energy-Efficient Neural Image Processing for Internet-of-Things Edge Devices," 60th IEEE International Midwest Symposium on Circuits and Systems (MWSCAS 2017), Aug. 2017.

[C13] **Jong Hwan Ko**, Josh Fromm, Shuayb Zarar, Matthai Philipose, Ivan Tashev, "Bit Precision Control of Deep Neural Networks for Efficient Speech Processing," Annual Conference on Neural Information Processing Systems (NIPS 2017), Machine Learning for Audio Signal Processing Workshop, Dec. 2017.

[C14] Taesik Na, **Jong Hwan Ko**, and Saibal Mukhopadhyay, "Adversarial Machine Learning Regularized with a Unified Embedding," Annual Conference on Neural Information Processing Systems (NIPS 2017), Machine Learning and Computer Security Workshop, Dec. 2017.

[C15] Saibal Mukhopadhyay, Marilyn Wolf, Mohammed F. Amir, Evan Gebahrdt, **Jong Hwan Ko**, Jae Ha Kung, and Burhan A. Mudassar, "The CAMEL Approach to Stacked Sensor Smart Cameras," Design, Automation, and Test in Europe (DATE 2018), Mar. 2018.

[C16] Taesik Na, **Jong Hwan Ko**, and Saibal Mukhopadhyay, "Cascade Adversarial Machine Learning Regularized with a Unified Embedding," International Conference on Learning Representations (ICLR 2018), Apr. 2018.

[C17] **Jong Hwan Ko**, Josh Fromm, Shuayb Zarar, Matthai Philipose, Ivan Tashev, "Limiting Numerical Precision of Neural Networks to Achieve Real-Time Voice Activity Detection," International Conference on Acoustic, Speech, and Signal Processing (ICASSP 2018), Apr. 2018.

[C18] Taesik Na, **Jong Hwan Ko**, and Saibal Mukhopadhyay, "Noise-Robust and Resolution-Invariant Image Classification with Pixel-Level Regularization," International Conference on Acoustic, Speech, and Signal Processing (ICASSP 2018), Apr. 2018.

[C19] Burhan Mudassar, **Jong Hwan Ko**, and Saibal Mukhopadhyay, "An Unsupervised Anomalous Event Detection Framework with Class Aware Source Separation," International Conference on Acoustic, Speech, and Signal Processing (ICASSP 2018), Apr. 2018.

[C20] Burhan A. Mudassar, **Jong Hwan Ko**, and Saibal Mukhopadhyay, "Edge-Cloud Collaborative Processing for Intelligent Internet of Things: A Case Study on Smart Surveillance," Design Automation Conference (DAC 2018), June 2018.

[C21] Mohammed F. Amir, **Jong Hwan Ko**, and S. Mukhopadhyay, "An Image Sensor SOC with Energy Harvesting Mixed-Vth Pixel Generating 5.8uW/mm2 Power Density and 0.77 Frames/second Self-Powered Frame Rate," IEEE SOI-3D-Subthreshold Microelectronics Technology Unified Conference (S3S 2018), Accepted for presentation.

[C22] Taesik Na, Burhan A. Mudassar, Priyabrata Saha, **Jong Hwan Ko**, and S. Mukhopad-

hyay, ”Adaptive Control of Camera Resolution with Robust Deep Learning Based Feedback for Object Tracking,” submitted to the Conference on Neural Information Processing Systems (NIPS 2018).

[C23] **Jong Hwan Ko**, Taesik Na, Mohammad Faisal Amir, and Saibal Mukhopadhyay, Edge-Host Partitioning of Deep Neural Networks with Feature Space Encoding for Resource-Constrained Internet-of-Things Platforms, submitted to the 15th IEEE International Conference on Advanced Video and Signal-based Surveillance (AVSS 2018).

REFERENCES

[1] A. Hampapur, L. Brown, J. Connell, S. Pankanti, A. Senior, and Y. Tian, "Smart surveillance applications, technologies and implications.pdf," in *ICICS-FCM*, 2003, ISBN: 0780381858.

[2] D. Grois and O. Hadar, "Complexity-Aware Adaptive Preprocessing Scheme for Region-of-Interest Spatial Scalable Video Coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 24, no. 6, pp. 1025–1039, 2014.

[3] M. Meddeb, M. Cagnazzo, and B. Pesquet-Popesc, "Region-of-Interest Based Rate Control Scheme for High Efficiency Video Coding," *APSIPA Transactions on Signal and Information Processing*, no. January, pp. 1–9, 2014.

[4] M. K. Law, a. Bermak, and C. Shi, "A low-power energy-harvesting logarithmic CMOS image sensor with reconfigurable resolution using two-level quantization scheme," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 58, no. 2, pp. 80–84, 2011.

[5] D. Shin, J. Lee, J. Lee, and H. J. Yoo, "DNPU: An 8.1TOPS/W reconfigurable CNN-RNN processor for general-purpose deep neural networks," *IEEE International Solid-State Circuits Conference*, vol. 60, pp. 240–241, 2017.

[6] G. Desoli, N. Chawla, T. Boesch, S. P. Singh, E. Guidetti, F. De Ambroggi, T. Majo, P. Zambotti, M. Ayodhyawasi, H. Singh, and N. Aggarwal, "A 2.9TOPS/W deep convolutional neural network SoC in FD-SOI 28nm for intelligent embedded systems," *IEEE International Solid-State Circuits Conference*, vol. 60, pp. 238–239, 2017.

[7] F. Conti, R. Schilling, P. D. Schiavone, A. Pullini, D. Rossi, F. K. Gurkaynak, M. Muehlberghuber, M. Gautschi, I. Loi, G. Haugou, S. Mangard, and L. Benini, "An IoT Endpoint System-on-Chip for Secure and Energy-Efficient Near-Sensor Analytics," *IEEE Transactions on Circuits and Systems I: Regular Papers*, pp. 1–14, 2017. arXiv: 1612.05974.

[8] A. Chefi, A. Soudani, and G. Sicard, "A CMOS image sensor with low-complexity video compression for wireless sensor networks," *IEEE NEWCAS*, pp. 1–4, 2013.

[9] M. Imran, N. Ahmad, K. Khursheed, M. A. Waheed, N. Lawal, and M. O'Nils, "Implementation of wireless vision sensor node with a lightweight bi-level video coding," *IEEE JETCAS*, vol. 3, no. 2, pp. 198–209, 2013.

[10] C. Kim and J.-n. Hwang, "Fast and automatic video object segmentation and tracking for content-based applications," *IEEE TCSVT*, vol. 12, no. 2, pp. 122–129, 2002.

[11] J. H. Ko, B. A. Mudassar, and S. Mukhopadhyay, "An Energy-Efficient Wireless Video Sensor Node for Moving Object Surveillance," *IEEE TMSCS*, vol. 1, no. 1, 2015.

[12] S. J. McKenna, Y. Raja, and S. Gong, "Tracking colour objects using adaptive mixture models," *Image and Vision Computing*, vol. 17, no. 3-4, pp. 225–231, 1999.

[13] K. Kim, T. H. Chalidabhongse, D. Harwood, and L. Davis, "Real-time foreground-background segmentation using codebook model," *Real-Time Imaging*, vol. 11, no. 3, pp. 172–185, 2005.

[14] A. Elgammal, D. Harwood, and L. Davis, "Non-parametric model for background subtraction," *ECCV*, vol. 1843, pp. 751–767, 2000.

[15] N. M. Oliver, B. Rosario, and A. P. Pentland, "A bayesian computer vision system for modeling human interactions," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 831–843, 2000.

[16] Y. Benezeth, P.-M. Jodoin, B. Emile, H. Laurent, and C. Rosenberger, "Comparative study of background subtraction algorithms," *Journal of Electronic Imaging*, vol. 19, no. 3, p. 033 003, 2010.

[17] N. Minegishi, J. Miyakoshi, S. Member, and Y. Kuroda, "VLSI Architecture Study of a Real-Time Scalable Optical Flow," *IEICE Transactions on Electronics*, no. 3, pp. 230–242, 2006.

[18] L. Li, W. Huang, I. Y. H. Gu, and Q. Tian, "Foreground object detection from videos containing complex background," *ACM MM*, vol. 03, p. 2, 2003.

[19] S. Lim and A. El Gamal, "Optical flow estimation using high frame rate sequences," *ICIP*, vol. 2, pp. 925–928, 2001.

[20] M.-c. Tuan and S.-L. Chen, "Fully Pipelined VLSI Architecture of a Real- Time Block-Based Object Detector for Intelligent Video Surveillance Systems," *2015 IEEE/ACIS 14th International Conference on Computer and Information Science (ICIS)*, pp. 149–154, 2015.

[21] W. Lai, X.-d. Gu, R.-h. Wang, L.-r. Dai, and H.-j. Zhang, "A region based multiple frame-rate tradeoff of video streaming," *2004 International Conference on Image Processing, 2004. ICIP '04.*, vol. 3, pp. 2067–2070, 2004.

[22] Y. Hu, F. Meng, and Y. Wang, "Improved JPEG Compression Algorithm Based on Saliency Maps," *CISP*, 2012.

[23] N. Tsapatsoulis, C. Loizou, and C. Pattichis, "Region of Interest Video Coding for Low Bit-Rate Transmission of Carotid Ultrasound Videos over 3G Wireless Networks," *Annual International Conference of the IEEE Engineering in Medicine and Biology*, pp. 3717–3720, 2007.

[24] D. Grois, S. Member, and O. Hadar, "Efficient Adaptive Bit-Rate Control for Scalable Video Coding by Using Computational Complexity-Rate- Distortion Analysis," *The IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*, 2011.

[25] N. A. Manap, G. Di Caterina, J. Soraghan, V. Sidharth, and H. Yao, "Smart surveillance system based on stereo matching algorithms with IP and PTZ cameras," *3DTV-CON 2010: The True Vision - Capture, Transmission and Display of 3D Video*, pp. 4–7, 2010.

[26] P. Panda, A. Sengupta, and K. Roy, "Conditional Deep Learning for Energy-Efficient and Enhanced Pattern Recognition Priyadarshini," *DATE*, 2016.

[27] J. K. Pavlo Molchanov, Xiaodong Yang , Shalini Gupta, Kihwan Kim, Stephen Tyree, "Online Detection and Classification of Dynamic Hand Gestures with Recurrent 3D Convolutional Neural Networks," *CVPR*, 2016.

[28] V. Nair and J. Clark, "An unsupervised, online learning framework for moving object detection," *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, vol. 2, pp. 3–10, 2004.

[29] H. Wang, X. Chai, Y. Zhou, and X. Chen, "Fast sign language recognition benefited from low rank approximation," *2015 11th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition, FG 2015*, 2015.

[30] Z. Dong, M. Pei, Y. He, T. Liu, Y. Dong, and Y. Jia, "Vehicle Type Classification Using Unsupervised Convolutional Neural Network," *2014 22nd International Conference on Pattern Recognition*, vol. 11, no. 4, pp. 172–177, 2014.

[31] S. Dodge and L. Karam, "A Study and Comparison of Human and Deep Learning Recognition Performance Under Visual Distortions," *arXiv preprint arXiv:1705.02498*, 2017. arXiv: 1705.02498.

[32] R. Geirhos, D. H. J. Janssen, H. H. Schütt, J. Rauber, M. Bethge, and F. A. Wichmann, "Comparing deep neural networks against humans: object recognition when

the signal gets weaker," *https://arxiv.org/abs/1706.06969*, 2017. arXiv: 1706.06969.

[33] Y. Zhou, S. Song, and N. M. Cheung, "On classification of distorted images with deep convolutional neural networks," *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 1213–1217, 2017. arXiv: 1701.01924.

[34] I. Vasiljevic, A. Chakrabarti, and G. Shakhnarovich, "Examining the Impact of Blur on Recognition by Convolutional Networks," *arXiv:1611.05760*, 2016. arXiv: 1611.05760.

[35] N. Das, M. Shanbhogue, S.-T. Chen, F. Hohman, L. Chen, M. E. Kounavis, and D. H. Chau, "Keeping the Bad Guys Out: Protecting and Vaccinating Deep Learning with JPEG Compression," *arXiv preprint arXiv:1705.02900*, 2017. arXiv: 1705.02900.

[36] I. Cevik, X. Huang, H. Yu, M. Yan, and S. Ay, "An Ultra-Low Power CMOS Image Sensor with On-Chip Energy Harvesting and Power Management Capability," *Sensors*, vol. 15, no. 3, pp. 5531–5554, 2015.

[37] G. Kim, Y. Lee, Z. Foo, P. Pannuto, Y. S. Kuo, B. Kempke, M. H. Ghaed, S. Bang, I. Lee, Y. Kim, S. Jeong, P. Dutta, D. Sylvester, and D. Blaauw, "A millimeter-scale wireless imaging system with continuous motion detection and energy harvesting," *IEEE Symposium on VLSI Circuits, Digest of Technical Papers*, no. December 2011, pp. 31–32, 2014.

[38] H.-t. Wang and W. D. Leon-Salas, "An Image Sensor With Joint Sensing and Energy Harvesting Functions," *IEEE Sensors Journal*, vol. 15, no. 2, pp. 902–916, 2015.

[39] S. K. Nayar, D. C. Sims, and M. Fridberg, "Towards self-powered cameras," in *2015 IEEE International Conference on Computational Photography (ICCP)*, 2015, pp. 1–10.

[40] A. Y.-c. Chiou and C.-c. Hsieh, "A 0 . 4V Self-Powered CMOS Imager with 140dB Dynamic Range and Energy Harvesting C86 C87," pp. 86–87, 2015.

[41] Y. P. Fallah, H. Mansour, and S. Khan, "A Link Adaptation Scheme for Efficient Transmission," *Circuits and Systems for Video Technology, IEEE*, vol. 18, no. 7, pp. 875–887, 2008.

[42] L Haratcherev and J. Taal, "Fast 802.11 link adaptation for real-time video streaming by cross-layer signaling," in *IEEE International Symposium on Circuits and Systems*, 2005.

[43] C Huang and C Lin, "Multiple-priority region-of-interest H.264 video compression using constraint variable bitrate control for video surveillance," *Optical Engineering*, vol. 48, no. 4, p. 047 004, 2009.

[44] M. Courbariaux, Y. Bengio, and J.-P. David, "Low Precision Storage for Deep Learning," *ICLR*, 2015. arXiv: 1412.7024.

[45] B. Moons, B. D. Brabandere, L. V. Gool, and M. Verhelst, "Energy-Efficient ConvNets Through Approximate Computing," in *IEEE Winter Conference on Applications of Computer Vision*, 2016. arXiv: arXiv:1603.06777v1.

[46] B. Moons and M. Verhelst, "An Energy-Efficient Precision-Scalable ConvNet Processor in 40-nm CMOS," *IEEE Journal of Solid-State Circuits*, vol. 52, no. 4, pp. 903–914, 2017.

[47] J. A. Hertz, A Krogh, and R. G. Palmer, *Introduction to the Theory of Neural Computation*. 1991.

[48] T. N. Sainath, B. Kingsbury, V. Sindhwani, E. Arisoy, and B. Ramabhadran, "Low-Rank Matrix Factorization for Deep Neural Network Training With High-Dimensional Output Targets," *ICASSP*, 2013.

[49] Y. H. Chen, T Krishna, J Emer, and V Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *ISSCC*, pp. 262–263, 2016.

[50] J. Koutník, F. Gomez, and J. Schmidhuber, "Evolving Neural Networks in Compressed Weight Space," *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation - GECCO '10*, p. 619, 2010.

[51] W. Chen, J. Wilcon, S. Tyree, K. Weinberger, and Y. Chen, "Compressing Convolutional Neural Networks in the Frequency Domain," *Gecco*, no. Nips, pp. 421–434, 2016. arXiv: arXiv:1602.05561v1.

[52] S. Han, H. Mao, and W. J. Dally, "Deep Compression - Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding," *ICLR*, 2016. arXiv: 1510.00149.

[53] M. Courbariaux and Y. Bengio, "BinaryNet: Training Deep Neural Networks with Weights and Activations Constrained to +1 or -1," *arXiv:1602.02830*, p. 9, 2016. arXiv: 1602.02830.

[54] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks," *Eccv*, pp. 1–17, 2016. arXiv: 1603.05279.

[55] J. Chung and T. Shin, "Simplifying Deep Neural Networks for Neuromorphic Architectures," in *DAC*, 2016, ISBN: 9781450342360.

[56] L. Wang, W. Ouyang, X. Wang, and H. Lu, "Visual tracking with fully convolutional networks," *Proceedings of the IEEE International Conference on Computer Vision*, vol. 11-18-Dece, pp. 3119–3127, 2016.

[57] M. Jaderberg, A. Vedaldi, and A. Zisserman, "Speeding up convolutional neural networks with low rank expansions," *CoRR*, vol. abs/1405.3866, 2014. arXiv: 1405.3866.

[58] M. Mathieu, M. Henaff, and Y. LeCun, "Fast training of convolutional networks through ffts," *CoRR*, vol. abs/1312.5851, 2013. arXiv: 1312.5851.

[59] N. Vasilache, J. Johnson, M. Mathieu, S. Chintala, S. Piantino, and Y. LeCun, "Fast convolutional nets with fbfft: A GPU performance evaluation," *CoRR*, vol. abs/1412.7580, 2014. arXiv: 1412.7580.

[60] Y. T. et al., "Vehicle detection and recognition for intelligent traffic surveillance system," in *Multimedia Tools and Applications*, vol. 76, 2017.

[61] G. C. et al., "Deep convolutional neural network based species recognition for wild animal monitoring," in *ICIP*, 2014.

[62] C. Z. et al., "Cross-scene crowd counting via deep convolutional neural networks," in *CVPR*, 2015.

[63] S. H. et al., "Deep Compression - Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding," *ICLR*, 2016. arXiv: 1510.00149.

[64] Y. H. C. et al., "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *JSSC*, 2017.

[65] K. B. et al., "14.6 a 0.62mw ultra-low-power convolutional-neural-network face-recognition processor and a cis integrated with always-on haar-like face detector," in *ISSCC*, 2017.

[66] S. T. et al., "Distributed deep neural networks over the cloud, the edge and end devices," in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, 2017, pp. 328–339.

[67] Z. C. et al., "Image quality assessment guided deep neural networks training," *CoRR*, vol. abs/1708.03880, 2017. arXiv: 1708.03880.

[68] Z. He and D. Wu, "Resource allocation and performance analysis of wireless video sensors," *Circuits and Systems for Video Technology, IEEE*, vol. 16, no. 5, pp. 590–599, 2006.

[69] A. a. El-Sherif, A. Mohamed, and V. C. M. Leung, "Optimum power and rate allocation in video sensor networks," *2013 IEEE Global Communications Conference (GLOBECOM)*, pp. 480–486, 2013.

[70] S. Pudlewski and T. Melodia, "Compressive Video Streaming: Design and Rate-Energy-Distortion Analysis," *IEEE Transactions on Multimedia*, vol. 15, no. 8, pp. 2072–2086, 2013.

[71] E. Soyak, S. A. Tsaftaris, and A. K. Katsaggelos, "Low-complexity tracking-aware H.264 video compression for transportation surveillance," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 21, no. 10, pp. 1378–1389, 2011.

[72] H. D. Cheng, X. H. Jiang, Y Sun, and J. L. Wang, "Color Image Segmentation : Advances & Prospects 1 INTRODUCTION," *Pattern Recognition*, 2001.

[73] Nordicsemi, "nRF24L01+ Single Chip 2.4 GHz Transceiver Preliminary Product Specification V1. 0," Tech. Rep.

[74] *The Open-Source H.264/AVC Verification Model*, `http://iphome.hhi.de/suehring/tml/`.

[75] Z. Wang, L. Lu, and A. C. Bovik, "Video quality assessment based on structural distortion measurement," *Signal Processing: Image Communication*, vol. 19, no. 2, pp. 121–132, 2004.

[76] M. F. Amir, "Design Mehodology for 3D-stacked Imaging Systems with Integrated Deep Learning," *Ph.D. Dissertation, Georgia Institute of Technology*, 2018.

[77] K. Z. Ahmed, M. F. Amir, J. H. Ko, and S. Mukhopadhyay, "Reconfigurable 96x128 Active Pixel Sensor with 2 . 1W / mm 2 Power Generation and Regulated Multi-Domain Power Delivery for Self-Powered Imaging," in *European Solid-State Circuit Conference (ESSCIRC)*, 2016, pp. 507–510, ISBN: 9781509029723.

[78] S. G. Narendra, L. C. Fujino, and K. C. Smith, "Through the looking glass? the 2015 edition: Trends in solid-state circuits from ISSCC," *IEEE Solid-State Circuits Magazine*, vol. 7, no. 1, pp. 14–24, 2015.

[79] S. Brutzer, B. Höferlin, and G. Heidemann, "Evaluation of background subtraction techniques for video surveillance," *IEEE CVPR*, pp. 1937–1944, 2011.

[80] W. Huang, K. Rajamani, M. R. Stan, and K. Skadron, "Scaling with design constraints: Predicting the future of big chips," *IEEE Micro*, vol. 31, no. 4, pp. 16–29, 2011.

[81] R. D. Gow, D. Renshaw, K. Findlater, L. Grant, S. J. McLeod, J. Hart, and R. L. Nicol, "A comprehensive tool for modeling cmos image-sensor-noise performance," *IEEE Transactions on Electron Devices*, vol. 54, no. 6, pp. 1321–1329, 2007.

[82] A. Shrivastava, Y. K. Ramadass, S. Khanna, S. Bartling, and B. H. Calhoun, "A 1.2w simo energy harvesting and power management unit with constant peak inductor current control achieving 83-92% efficiency across wide input and output voltages," in *2014 Symposium on VLSI Circuits Digest of Technical Papers*, 2014, pp. 1–2.

[83] M. B. Dillencourt, H. Samet, and M. Tamminen, "A general approach to connected-component labeling for arbitrary image representations," *Journal of the ACM*, vol. 39, no. 2, pp. 253–280, 1992.

[84] N. Goyette, P. M. Jodoin, F. Porikli, J. Konrad, and P. Ishwar, "changedetection . net : A New Change Detection Benchmark Dataset," *IEEE CVPR*, 2012.

[85] M. Genovese and E. Napoli, "ASIC and FPGA implementation of the gaussian mixture model algorithm for real-time segmentation of high definition video," *IEEE TVLSI*, vol. 22, no. 3, pp. 537–547, 2014.

[86] Y. Emre and C. Chakrabarti, "Energy and quality-aware multimedia signal processing," *IEEE Transactions on Multimedia*, vol. 15, no. 7, pp. 1579–1593, 2013.

[87] H. C. Chang, J. W. Chen, B. T. Wu, C. L. Su, J. S. Wang, and J. I. Guo, "A dynamic quality-adjustable H.264 video encoder for power-aware video applications," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 19, no. 12, pp. 1739–1754, 2009.

[88] N. Goyette, P. M. Jodoin, F. Porikli, J. Konrad, and P. Ishwar, "changedetection.net: A new change detection benchmark dataset," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 2012.

[89] A. Hampapur, L. Brown, J. Connell, a. Ekin, N. Haas, M. Lu, H. Merkl, and S. Pankanti, "Smart video surveillance: exploring the concept of multiscale spatiotemporal tracking," *IEEE Signal Processing Magazine*, vol. 22, no. 2, pp. 38–51, 2005.

[90] S Zhang, S. C. Chan, R. D. Qiu, K. T. Ng, Y. S. Hung, and W Lu, "On The Design And Implementation of a High Definition Multiview Intelligent Video Surveillance System," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 2013.

[91] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," *Advances In Neural Information Processing Systems*, pp. 1–9, 2012. arXiv: 1102.0183.

[92] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 815–823.

[93] Y. LeCun, K. Kavukcuoglu, and C. Farabet, "Convolutional networks and applications in vision," *IEEE International Symposium on Circuits and Systems: Nano-Bio Circuit Fabrics and Systems*, 2010.

[94] J. Dai, Y. Li, K. He, and J. Sun, "R-FCN: Object Detection via Region-based Fully Convolutional Networks," *Nips*, 2016. arXiv: 1605.06409.

[95] Y. Jia, *Caffe: An open source convolutional architecture for fast feature embedding*. 2014.

[96] Jia Deng, Wei Dong, R. Socher, Li-Jia Li, Kai Li, and Li Fei-Fei, "ImageNet: A large-scale hierarchical image database," *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009.

[97] YuwenXiong, *R-FCN with joint training and python support*.

[98] T. Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common objects in context," *ECCV*, 2014. arXiv: 1405.0312.

[99] S. Wu, S. Zhong, and Y. Liu, "Deep residual learning for image recognition," in *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, 2017, ISBN: 978-1-4673-6964-0. arXiv: 1512.03385.

[100] W. Luo, J. Huang, and G. Qiu, "JPEG Error Analysis and Its Applications to Digital Image Forensics," *IEEE Transactions on Information Forensics and Security*, vol. 5, no. 3, pp. 480–491, 2010.

[101] A. Katsaggelos and Y. Eisenberg, "Advances in efficient resource allocation for packet-based real-time video transmission," *Proceedings of the IEEE*, vol. 93, no. 1, pp. 135–147, 2005.

[102] Q. Wang, M. Hempstead, and W. Yang, "A Realistic Power Consumption Model for Wireless Sensor Network Devices," *2006 3rd Annual IEEE Communications Society on Sensor and Ad Hoc Communications and Networks*, pp. 286–295, 2006.

[103] S. Pudlewski and T. Melodia, "A Rate-Energy-Distortion Analysis for Compressed-Sensing-Enabled Wireless Video Streaming on Multimedia Sensors," *2011 IEEE Global Telecommunications Conference - GLOBECOM 2011*, pp. 1–6, 2011.

[104] ITU-T Recommendation H.264, "Advanced Video Coding for Generic Audiovisual Services," Tech. Rep., 2005.

[105] Y. Sun, I. Ahmad, and S. Member, "A Robust and Adaptive Rate Control Algorithm for Object-Based Video Coding," *IEEE Transactions on Circuits and Systems*, vol. 14, no. 10, pp. 1167–1182, 2004.

[106] Y. Z. et al., "PID-Based Bit Allocation Strategy for H.264/AVC Rate Control," *IEEE Transactions on Circuits and Systems-II: Express Briefs*, vol. 58, no. 3, pp. 184–188, 2011.

[107] Z. Chen and K. N. Ngan, "Towards rate-distortion tradeoff in real-time color video coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 2, pp. 158–167, 2007.

[108] Y. Hou, P. Wang, W. Xiang, Z. Gao, and C. Hou, "A novel rate control algorithm for video coding based on fuzzy-PID controller," *Signal, Image and Video Processing*, 2013.

[109] S. Mirsamadi and I. Tashev, "Causal speech enhancement combining data-driven learning and suppression rule estimation," *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, vol. 08-12-Sept, pp. 2870–2874, 2016.

[110] J. L. Ticknor, "A Bayesian regularized artificial neural network for stock market forecasting," *Expert Systems with Applications*, vol. 40, no. 14, pp. 5501–5506, 2013.

[111] T. Na et al., "Speeding Up Convolutional Neural Network Training with Dynamic Precision Scaling and Flexible Multiplier-Accumulator," *ISLPED 2016*,

[112] *MNIST database*, `http://yann.lecun.com/exdb/mnist/`.

[113] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-Based Learning Applied to Document Recognition," *Proceedings of IEEE*, 1998.

[114] *DDR3 SDRAM, JESD79-3F*, `http://www.jedec.org/`.

[115] *SD Specifications Version 4.10*, SD Association.

[116] D. Kim, J. Kung, S. Chai, S. Yalamanchili, and S. Mukhopadhyay, "Neurocube : A Programmable Digital Neuromorphic Architecture with High-Density 3D Memory," in *ISCA*, 2016, ISBN: 978-1-4673-8947-1.

[117] J. Kung, D. Kim, and S. Mukhopadhyay, "Dynamic Approximation with Feedback Control for Energy-Efficient Recurrent Neural Network Hardware," *ISLPED*, 2016.

[118] E. Azarkhish, D. Rossi, I. Loi, and L. Benini, "Neurostream: Scalable and Energy Efficient Deep Learning with Smart Memory Cubes," pp. 1–14, 2017. arXiv: 1701.06420.

[119] M. Gao and M. Horowitz, "TETRIS : Scalable and Efficient Neural Network Acceleration with 3D Memory," *Asplos*, pp. 751–764, 2017.

[120] *Digital compression and coding of continuous-tone still images: JPEG File Interchange Format (JFIF)*.

[121] N. Tavakoli, "Entropy and Image Compression," *Journal of Visual Communication and Image Representation*, 1993.

[122] J. Kung, D. Kim, and S. Mukhopadhyay, "A power-aware digital feedforward neural network platform with backpropagation driven approximate synapses," *ISLPED*, 2015.

[123] F. Wu, *Advances in Visual Data Compression and Communication: Meeting the Requirements of New Applications*. 2014.

[124] *CNAE-9 Dataset*, https://archive.ics.uci.edu/ml/datasets/CNAE-9.

[125] K. T. Malladi, "Towards energy proportional datacenter memory with mobile dram," *ACM SIGARCH Computer Architecture News*, 2012.

[126] Micron, "Micron 2Gb: x4, x8, x16 DDR3 SDRAM. Data Sheet MT41J128M16HA-125," Tech. Rep., 2010.

[127] *Bus speed specification, SD Association*, https://www.sdcard.org/developers/overview/bus_speed/index.html.

[128] R. G. Lyons, *Understanding Digital Signal Processing*.

[129] O. Rippel, J. Snoek, and R. P. Adams, "Spectral representations for convolutional neural networks," NIPS'15, pp. 2449–2457, 2015.

[130] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," ICML'15, pp. 448–456, 2015.

[131] A. Krizhevsky et al., "Learning multiple layers of features from tiny images," in *Computer Science Department, University of Toronto*, 2009.

[132] P. Milder et al., *Computer Generation of Hardware for Linear Digital Signal Processing Transforms*. 2012, ISBN: 0001411101.

[133] J. H. Ko, D. Kim, T. Na, J. Kung, and S. Mukhopadhyay, "Adaptive Weight Compression for Memory-Efficient Neural Networks," in *Design, Automation, and Test in Europe (DATE)*, 2017.

[134] *Jedec standard: Low power double data rate 4*.

[135] S. H. et al., "Eie: Efficient inference engine on compressed deep neural network," in *ISCA*, ser. ISCA '16, Seoul, Republic of Korea, 2016, ISBN: 978-1-4673-8947-1.

[136] R. S. RS9110-N-11-02 802.11bgn WLAN Module Data Sheet.

[137] K. S. et al., "Very deep convolutional networks for large-scale image recognition," *CoRR*, 2014. arXiv: 1409.1556.

[138] S. W. et al., "Deep residual learning for image recognition," in *CVPR*, 2017, ISBN: 978-1-4673-6964-0. arXiv: 1512.03385.